NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

X3D-EARTH: FULL GLOBE COVERAGE UTILIZING MULTIPLE DATASETS

by

Dale R. Tourtelotte

September 2010

Thesis Advisor: Don Brutzman
Second Readers: Byounghyun Yoo
Don McGregor

This thesis was done at the MOVES Institute
Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2010	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE X3D-Earth: Full Globe Coverage Utilizing Multiple Datasets **6. AUTHOR** Dale R. Tourtelotte	**5. FUNDING NUMBERS**
7. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME AND ADDRESS Navy Modeling and Simulation Office (NMSO)	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol Number: ___N/A__.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	**12b. DISTRIBUTION CODE** A

13. ABSTRACT

U.S. Armed Forces are typically utilizing and paying for commercial proprietary licensed products to visualize geospatial scenes, which nevertheless are themselves derived from government-produced data. This is unsatisfactory. This thesis has developed an open-source, royalty-free method for generating full-coverage 3D globes using the Extensible 3D (X3D) Graphics International Standard. Specifically, this thesis designs and generates robust globe models developing an instance of full global coverage utilizing X3D-Earth. In order to show interoperability and "Mash up" capabilities, multiple formats are used, including DTED Level 0 and NGA-produced satellite imagery. Imagery and corresponding terrain datasets are preprocessed using image processing and terrain parsing software, creating the X3D-Earth quad-tree tiles into multiple level-of-detail (LOD) file archives. Finally, these pyramidal, locale scenes are grouped and connected to form an overall X3D-Earth globe. Preprocessing, processing and data storage are performed with the NPS Hamming Supercomputer. The result of this work is a methodology for generating X3D-Earth locales that is suitable for massive replication, optimization and reuse. Current results are promising and further work is warranted. The ultimate product is expected to further enable new tactical capabilities, provide direct end-user control of visualization-data pedigrees, and enable improved operational and navigational situational awareness for all deployed warfighters.

14. SUBJECT TERMS X3D, X3D-Earth, geospatial, globe, super computer, imagery , terrain datasets	**15. NUMBER OF PAGES** 165
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

THIS PAGE INTENTIONALLY LEFT BLANK

This thesis was done at the MOVES Institute
Approved for public release; distribution is unlimited

X3D-EARTH: FULL GLOBE COVERAGE UTILIZING MULTIPLE DATASETS

Dale R. Tourtelotte
Lieutenant, United States Navy
B.A., Texas A & M University, 2005

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN
MODELING, VIRTUAL ENVIRONMENTS, AND SIMULATION (MOVES)

from the

NAVAL POSTGRADUATE SCHOOL
September 2010

Author: Dale R. Tourtelotte

Approved by: Don Brutzman, PhD
Thesis Advisor

Don McGregor
Second Reader

Byounghyun Yoo, PhD
Second Reader

Mathias Kolsch, PhD
Chair, MOVES Academic Committee

Peter J. Denning, PhD
Chair, Computer Science Department

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

U.S. Armed Forces are typically utilizing and paying for commercial proprietary licensed products to visualize geospatial scenes, which nevertheless are themselves derived from government-produced data. This is unsatisfactory. This thesis has developed an open-source, royalty-free method for generating full-coverage 3D globes using the Extensible 3D (X3D) Graphics International Standard. Specifically, this thesis designs and generates robust globe models developing an instance of full global coverage utilizing X3D-Earth. In order to show interoperability and “Mash up” capabilities, multiple formats are used, including DTED Level 0 and NGA-produced satellite imagery. Imagery and corresponding terrain datasets are preprocessed using image processing and terrain parsing software, creating the X3D-Earth quad-tree tiles into multiple level-of-detail (LOD) file archives. Finally, these pyramidal, locale scenes are grouped and connected to form an overall X3D-Earth globe. Preprocessing, processing and data storage are performed with the NPS Hamming Supercomputer. The result of this work is a methodology for generating X3D-Earth locales that is suitable for massive replication, optimization and reuse. Current results are promising and further work is warranted. The ultimate product is expected to further enable new tactical capabilities, provide direct end-user control of visualization-data pedigrees, and enable improved operational and navigational situational awareness for all deployed warfighters.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ASCII	American Standard Code for Information Interchange test-based file format
CENIC	Corporation for Education Network Initiatives in California
CIB	Controlled Imagery Base image dataset
DNC	Digital Nautical Chart
DTED	Digital Terrain Elevation Data
DTED0	DTED Level 0, 1 kilometer resolution terrain data
DTED1	DTED Level 1, 100 meter resolution terrain data
DTED2	DTED Level 2, 30 meter resolution terrain data
FOUO	For Official Use Only
GeoLOD	Geographically referenced Level of Detail (X3D Node)
GeoTIFF	Geographically referenced TIFF image format
GUI	Graphical User Interface
IDE	Integrated Development Environment
JPEG	Joint Photographic Experts Group – Image format
NGA	National Geospatial-Intelligence Agency
NPS	Naval Postgraduate School
SAVAGE	Scenario Authoring and Visualization for Advanced Graphical Environments Research Group
SCP	Secure Copy Protocol
SRTM	Shuttle Radar Topography Mission (Terrain Data Type)
TIFF	Tagged Image File Format
VRML	Virtual Reality Modeling Language
X3D	Extensible 3D Graphics
XYZ	Georeferenced file format that represents Cartesian coordinates

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

We gratefully acknowledge the Navy Modeling and Simulation Office (NMSO) and director John Moore for project support.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The primary goal of this thesis is to investigate and design a general method for generating a robust globe model, with full global coverage utilizing X3D-Earth. The emphasis is on using unclassified datasets readily available to United States military, specifically Digital Nautical Chart (DNC), Digital Terrain Elevation Data (DTED), and National Geospatial-Intelligence Agency (NGA) satellite imagery. DTED Level 0 data is 1-kilometer-resolution terrain data and is open for general use. DTED Level 1 and 2 data, 100-meter and 30-meter resolution, respectively, are unclassified but releasable only to U.S. government agencies and are For Official Use Only (FOUO). There is no FOUO data in this thesis; document and release is unrestricted. The DNC data also is releasable only to U.S government agencies and is FOUO. The result of the process is envisioned to be a robust methodology for generating X3D-Earth locales for widespread replication and reuse. The ultimate product will be the full X3D-Earth Model that preprocesses multiple datasets for use in U.S. Navy navigational and operational situational awareness.

A. MOTIVATION

As a former Navigation Officer, the author realizes the power of having a full 3D environment in navigation, situational awareness, and maritime operations. The primary resource for any Surface Warfare Officer to visualize an operational situation is through 2D means or, at best, static, real world models. This thesis is intended to develop the capability to render the operational picture in 3D. X3D-Earth is an excellent asset for such an application. X3D-Earth programs used in this thesis are open source, partly developed at the Naval Postgraduate School (NPS), resulting in complete access to learning materials, code base, and subject-matter experts, making this the ideal research platform for this thesis. Furthermore, access to the NPS Hamming Supercomputer for preprocessing of the various dataset sets lends another great opportunity for this thesis research.

B. PROBLEM STATEMENT

The primary research question deals directly with a full 3D representation of the world using multiple datasets. U.S. Armed Forces are typically utilizing and paying for commercial proprietary licensed products for visualizing geospatial scenes, which nevertheless are themselves derived from government produced data. The goal of this thesis is to develop an open-source, royalty-free method for generating full-coverage 3D globes using the Extensible 3D (X3D) Graphics International Standard. In order to show interoperability and "Mashup" capabilities, multiple formats are used, starting with DTED Level 0 and NGA satellite imagery. This thesis utilizes an open-source elevation parsing and image tiling program called Rez, developed by Dr. Chris Thorne, Managing Director of VRShed. Processing and data storage is accomplished with the Hamming Supercomputer located at NPS, commonly referred to as Hamming. The supercomputer utilizes Sun Grid Engine for job scheduling, and is fully compatible of scheduling the Java-based Rez program for processing the elevation data and imagery. This work is expected to eventually further enable new tactical capabilities, direct control of visualization-data pedigrees, and improved situational awareness for all deployed warfighters.

C. METHODOLOGY

There are many sources of elevation data available. The large size of these datasets inhibits innovation, and to date, only commercial companies are the primary producers of full globe coverage using 3D elevation data. With Hamming's capabilities at NPS and less expensive storage costs, it is possible to produce 3D globes of increasing fidelity that are tested and evaluated at each step along the way.

The specific methodology for developing a full globe of X3D-Earth tile sets is to first obtain the data from NGA via large-capacity external hard drives. Once the data is obtained, it is transferred to the Hamming file system for processing and storage. Rez is utilized for the elevation data and image tiling production process. Rez is designed to process one or multiple elevation files via arguments on the command line. A program is developed to repeatedly invoke Rez for tiling of the elevation data and imagery, and then this program sends the Java execution calls to the Sun Grid Engine on the Hamming Supercomputer. Once this processing is complete, another Java program developed with this thesis is executed to build the parent X3D scenes that bring all the preprocessed tile sets together as interlinked X3D scenes to create a full 3D globe.

This process is a computationally intensive task when dealing with the entire globe; therefore, the datasets are first reduced to the coverage of California as a demonstration of each step in the process. Once a stable demonstration is achieved, the process is then scaled up to include the data for the entire globe.

D. THESIS ORGANIZATION

This thesis is organized into nine additional chapters. Chapter II describes the related work for this thesis, i.e., Google Earth, NASA World Wind, and BING Maps Platform. Chapter III describes the prior work in Web-based geospatial modeling and the origins of X3D-Earth. Chapter IV describes the datasets used and/or examined during the process of this thesis. Chapter V describes the X3D Geospatial Component and the X3D-Earth distributed scene graph. Chapter VI contains the detailed problem description and detailed architecture of this thesis, including the approach, lessons learned and descriptions of the programs created in this thesis process. Chapter VII describes how the data was obtained, generated, stored, and externally exposed. Chapter VIII describes exemplar applications examined, used, and/or created in this thesis process. Chapter IX describes the experimental results of this thesis. Chapter X contains the conclusions and recommendations for future works.

THIS PAGE INTENTIONALLY LEFT BLANK

II. RELATED WORK

A. INTRODUCTION

There are pre-existing government and commercial 3D globe models. They are discussed in this section.

B. GOOGLE EARTH

Google Earth is proprietary software that allows a user to view the globe in a full 3D environment. “Google Earth lets you fly anywhere on Earth to view satellite imagery, maps, terrain, 3D buildings, from galaxies in outer space to the canyons of the ocean. You can explore rich geographical content, save your toured places, and share with others” (Google, 2010). A detailed description of the history of Google Earth can be found in Appendix G.

1. Features and Capabilities

The features of Google Earth include a search engine for locating large amounts of content for the entire globe. Also, the user can explore the moon, Mars, historical imagery, and the ocean (Google, 2010b). Viewing Earth in 3D is at its best in Google Earth. The technology that Google utilizes in its Google Earth 5 program is top of the line. Some of the features are illustrated in Figures 1, 2, 3, and 4. Figure 1 illustrates a relatively new feature in which the user can view U.S.-based flights. Figure 2 illustrates 3D building tours of U.S. capitol buildings. The other two figures are similar; Figure 3 illustrates an example of 3D Cathedrals, and Figure 4 illustrates a 3D model of a University building. These buildings illustrate the capabilities of Google Earth and some of the work done in 3D visualization.

2. Data Sources

The sources of data and commercial imagery for Google Earth are extensive and come from U.S. and foreign governments at the federal, state, and local level. The terrain data used by Google is typically Shuttle Radar Topography Mission (SRTM) data provided by the U.S. Government. The imagery used in Google Earth is a more complicated matter. The list of all image sources is difficult to find, but the list on the Google Earth Wikipedia page has a list of the following sources for imagery:

- DigitalGlobe — the provider of high resolution imagery to Google Earth
- EarthSat
- GeoEye-1 (ORBVIEW-3's successor)
- GlobeXplorer
- IKONOS Satellite (ORBVIEW-2 is successor)
- Pictometry
- Spot Image
- TerraLook
- ViewGL - updated aerial imagery for Google Earth
- CNES

Additionally, a great deal of imagery is provided by local, state, and national governments. The resolutions of images vary depending on the location and source. Much of the globe is 15m resolution at a minimum, but can get to 0.1 meter resolution in urban areas.

For a full description of Google Earth and its capabilities, go to http://earth.google.com.

3. KML (Placemark) Standard

Formerly Keyhole Markup Language, KML has been used by Google as its language of choice for displaying geospatial data on its Earth. Google submitted the language to Open Geospatial Consortium to be adapted and adopted as a standard. The result is KML Version 2.2 as an OGC implementation standard. There are four objectives for the KML standards work (the following bullets were taken from OGC, 2010):

- That there be one international standard language for expressing geographic annotation and visualization on existing or future Web-based online and mobile maps (2D) and earth browsers (3D).
- That KML be aligned with international best practices and standards, thereby enabling greater uptake and interoperability of earth browser implementations.

- That the OGC and Google will work collaboratively to ensure that the KML implementer community is properly engaged in the process and that the KML community is kept informed of progress and issues.
- That the OGC process will be used to ensure proper life-cycle management of the KML Standard, including such issues as backwards compatibility.

KML is an XML-based language that focuses on geographic visualization, including the representation of visual data, maps and images, and the user's navigation of the 3D world (OGC, 2010).

A full copy of a reference for KML 2.2 can be found at http://portal.opengeospatial.org/files/?artifact_id=23689.

The schema for KML 2.2 can be found at http://schemas.opengis.net/kml.

For further information on the KML Standard, go to http://www.opengeospatial.org/standards/kml.

4. License

The full license agreement for Google Earth (free version) can be found at http://earth.google.com/intl/en-US/license.html.

A summary of the license agreement is that Google Earth can be used for personal use only. The content cannot be used or accessed by another person. All content can only be viewed and not downloaded except through screenshots. The user cannot access the actual imagery and data.

The restrictiveness of the default license against official use of Google Earth are a significant impediment to proper usage by governments, universities, etc. This commercial barrier is especially puzzling since the overwhelming bulk of available imagery is funded by numerous governments worldwide, and since many of the technologies integrated by this tool were initially developed through university research.

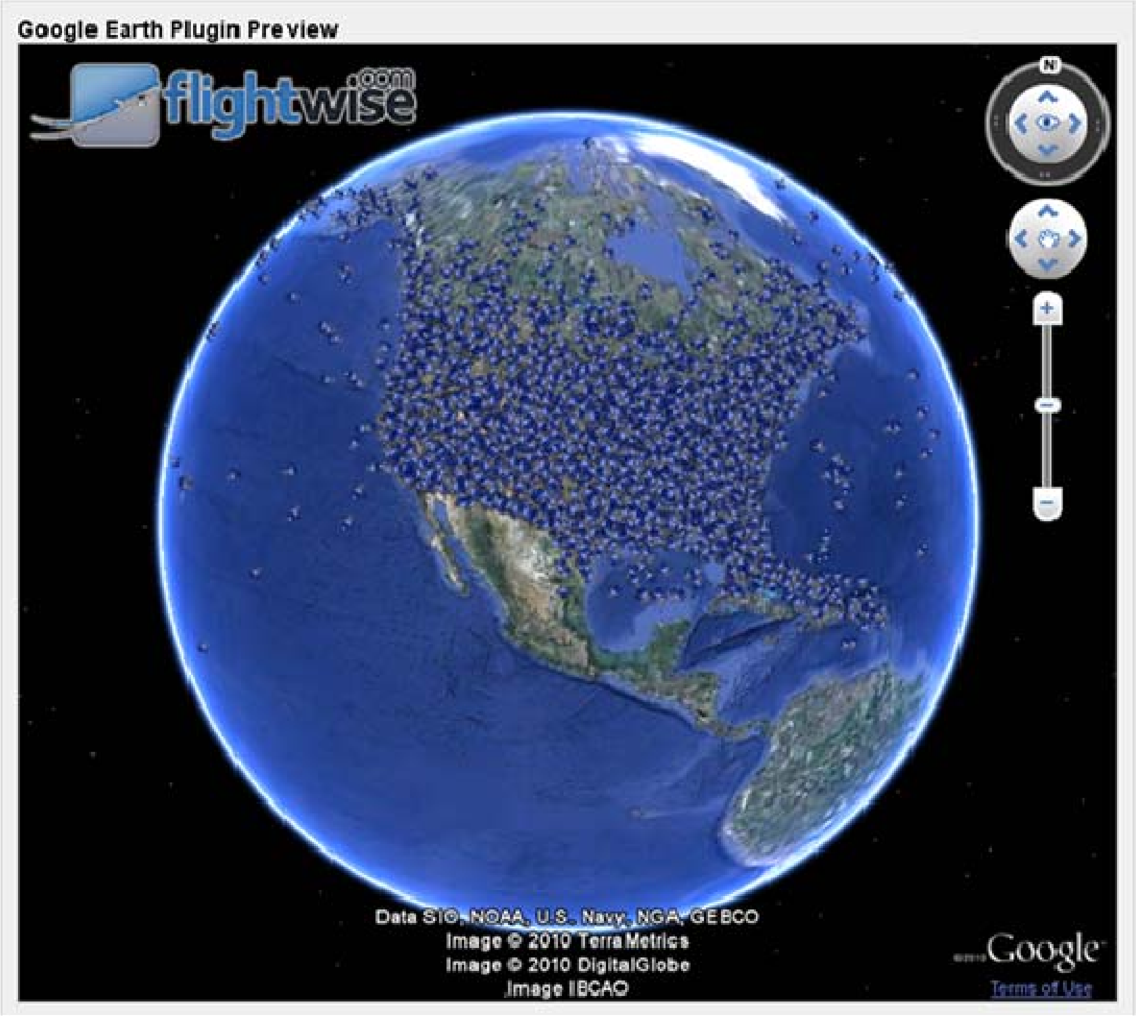

Google Earth Plugin Preview
flightwise.com
N
Data SIO, NOAA, U.S. Navy, NGA, GEBCO
Image © 2010 TerraMetrics
Image © 2010 DigitalGlobe
Image IBCAO
Google
Terms of Use

Figure 1. Hourly Snapshot of Active Airline Flights

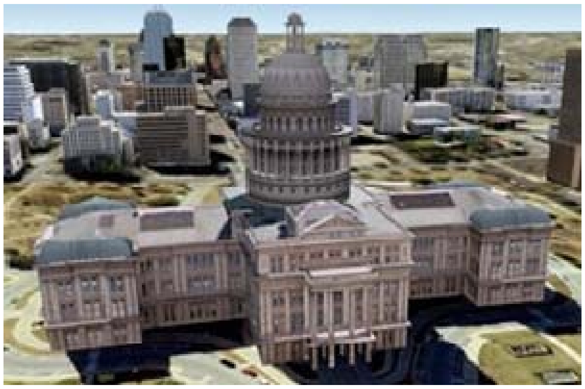


Figure 2. 3D Model of U.S. Capitol Building (From Google Earth Gallery, 2010)

Figure 3. Google Earth: 3D Model of a Cathedral (From Google, 2010a)

Figure 4. Google Earth: 3D Model of a University (From Google, 2010a)

C. NASA WORLD WIND

1. Features and Capabilities

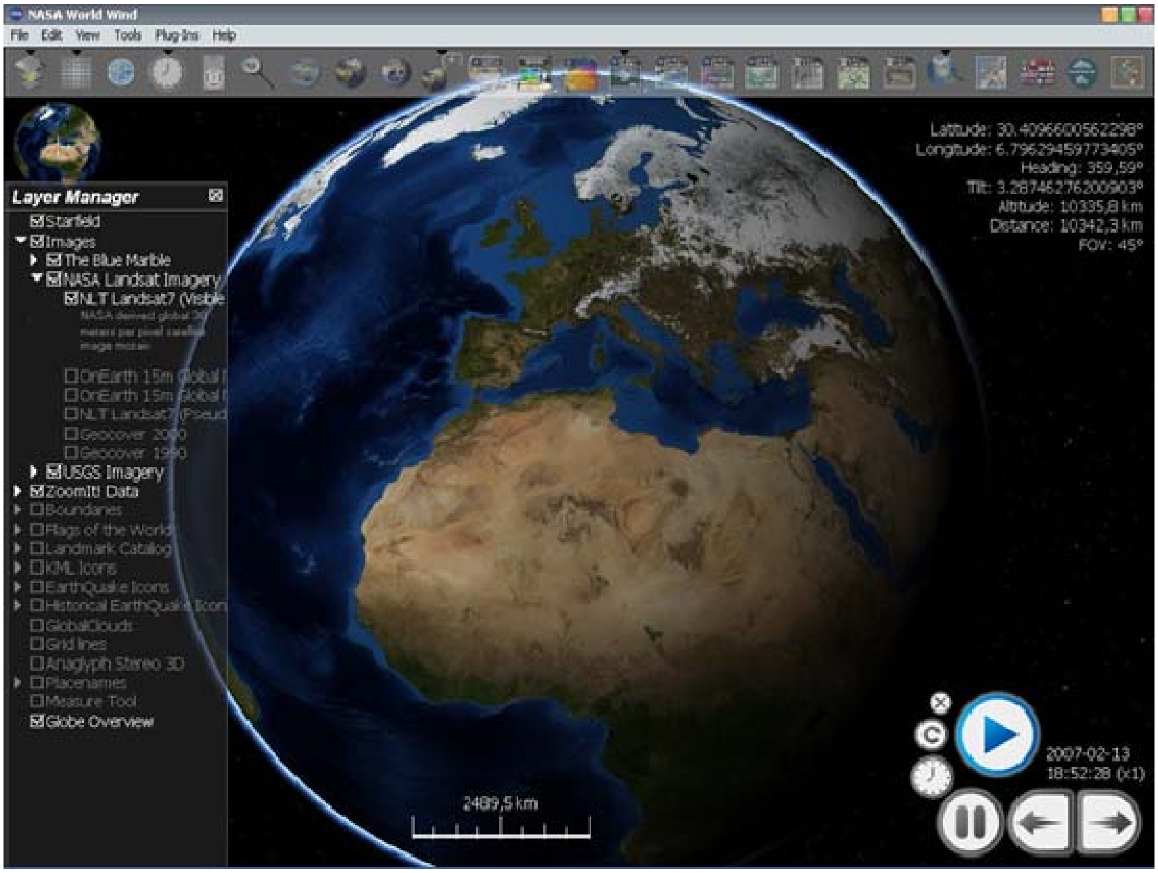
This section covers the history of NASA World Wind and the sources of data. The World Wind Wiki describes the software as:

> At its simplest you can think of World Wind as a desk globe, however World Wind is not a simple desk globe.
>
> World Wind allows any user to zoom from outer space into any place on Earth. World Wind uses satellite imagery and elevation data to allow users to experience Earth terrain in visually rich 3D, just as if they were really there. Virtually visit anyplace in the world. Look across the Andes, into the Grand Canyon, over the Alps or along the African Sahara.

> Unlike your desk globe World Wind can display thousands of place names all over the world, from country capitals to villages in sparsely populated regions. You can see country borders, and in some cases intracountry borders such as U.S. states.

NASA World Wind is a 3D earth viewer that allows the user to view, search, and placemark the earth. It also has the capability to view the Moon, Mars, Venus, Jupiter, and “the sky with the stars and galaxies” (NASA, 2010). Some of the tools that are available for use with World Wind are Rapid Fire MODIS, and WMS Browser. Rapid Fire MODIS is a tool for obtaining near real-time images from satellites and projecting them onto the globe. These images are up to 250 meters resolution. WMS Browser is a tool for connecting nearly any WMS map server to World Wind and enables the user to project these onto the globe (NASA, 2010). “Although NASA’s World Wind for Java provides an advanced system architecture implemented in an open-source code base, it does not provide generalized simulation functionality and requires Java developers to embed World Wind in their own applications” (Yoo & Brutzman, 2009).

Figure 5 is a snapshot of the interface and shows the Blue Marble imagery.



Layer Manager

- ☑ Starfield
- ☑ Images
  - ☑ The Blue Marble
  - ☑ NASA Landsat Imagery
    - ☑ NLT Landsat7 (Visible
      NASA derived global 30
      meters per pixel satellit
      image mosaic
    - ☐ OnEarth 15m Global
    - ☐ OnEarth 15m Global
    - ☐ NLT Landsat7 (Pseud
    - ☐ Geocover 2000
    - ☐ Geocover 1990
  - ☑ USGS Imagery
- ☑ ZoomIt! Data
- ☐ Boundaries
- ☐ Flags of the World
- ☐ Landmark Catalog
- ☐ KML Icons
- ☐ EarthQuake Icons
- ☐ Historical EarthQuake Icon
- ☐ GlobalClouds
- ☐ Grid lines
- ☐ Anaglyph Stereo 3D
- ☐ Placenames
- ☐ Measure Tool
- ☑ Globe Overview

Latitude: 30.4096600562298°
Longitude: 6.79629459773405°
Heading: 359,59°
Tilt: 3.28746276200903°
Altitude: 10335,8 km
Distance: 10342,3 km
FOV: 45°

2007-02-13
18:52:28 (x1)

Figure 5. Snapshot of World Wind With Blue Marble Imagery. Image available from World Wind Central, http://worldwindcentral.com/wiki/World_Wind.

2. Source Data

The source data used in World Wind is less extensive than that of Google Earth, and it comes from U.S. government sources only. The terrain data is SRTM data at 30 meters resolution (NASA, 2010). The imagery is Blue Marble Next Generation (500 meters), Landsat 7 (15m), and USGS data (up to 0.25 meters) (NASA, 2010). World Wind also uses a ZoomIt plug-in, which has high-resolution data for many U.S. locations and some international places. For a detailed list of these locations and sources, go to http://worldwindcentral.com/wiki/Add-on:ZoomIt!.

Figure 6 is a snapshot of the use of SRTM terrain data and Landsat 7 imagery.

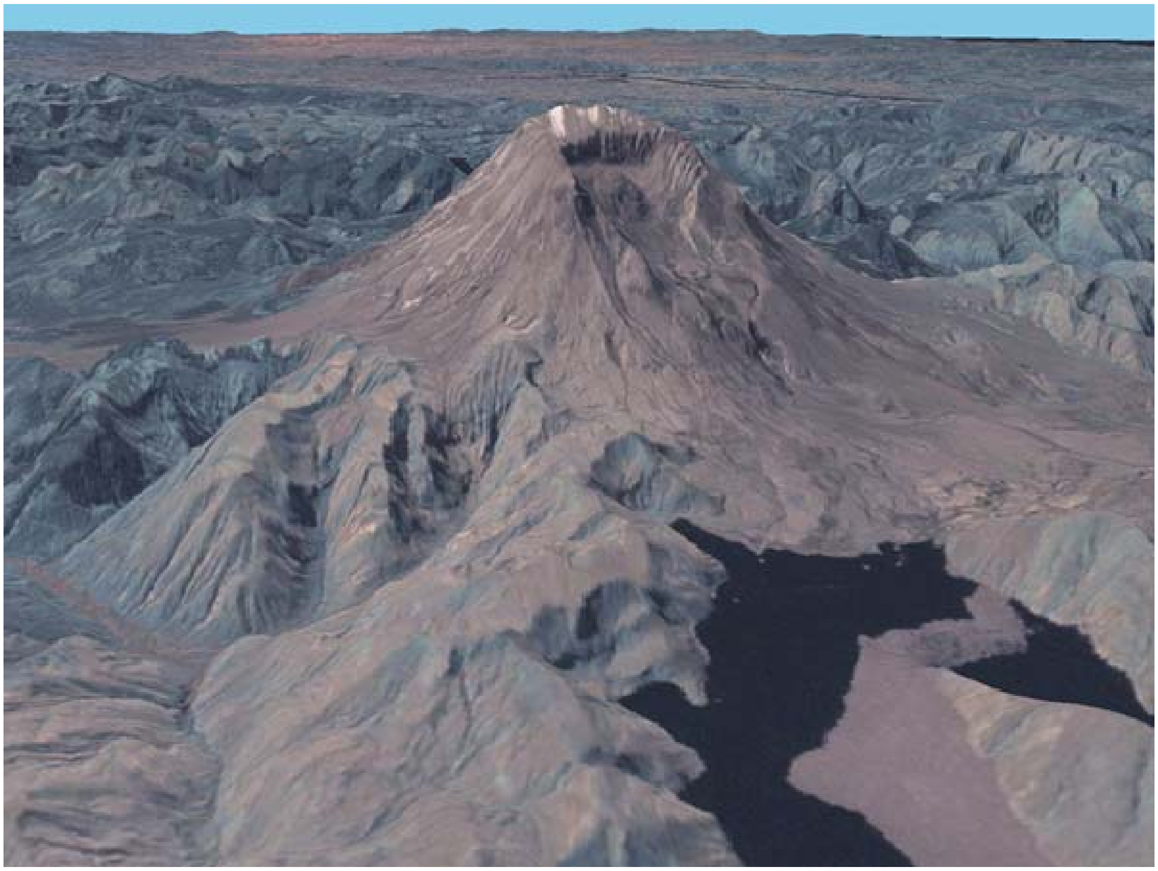


Figure 6. NASA World Wind: SRTM Terrain Data and LandSat 7 Imagery (From NASA, 2010)

3. License

The full license agreement for NASA World Wind can be found at http://worldwind.arc.nasa.gov/worldwind-nosa-1.3.html.

A summary of the license agreement is that NASA World Wind is an open-source program that can be distributed, redistributed, modified, etc. The software must be accompanied by the license agreement and is copyright of the U.S. Government. This also requires that anyone using the software must register it when using and/or redistributing.

D. BING MAPS PLATFORM (FORMERLY MICROSOFT VIRTUAL EARTH)

1. Features and Capabilities

This section describes the features and capabilities of Bing Maps.

Bing Maps Platform is a Microsoft-created 3D globe that offers a number of features. It was originally called Microsoft Virtual Earth. The features, described on the Overview Web site (BING, 2010), include:

- Multiple imagery sets from overhead satellite to street level views to a bird's eye view which allows the user to view from a 45-degree angle from four different directions (north, south, east, and west).
- Overlays of standard and custom data.
- Developer support options available, including forums, trusted by thousands of organizations worldwide.
- Multiple Applications for developers to expand upon.

The primary purpose of the Bing Maps Platform is for business use. The design and purpose of the software is to be integrated into a user's or developer's Web site. This allows the user to display data on the globe for operational purposes. Figures 7 and 8 illustrate the use of Bing Maps Platform in the commercial sector. Coldwell Banker Commercial, Figure 7, uses the software to display its properties for sale throughout the U.S. and the globe. Eye On Earth, Figure 8, uses the software to display statistics about the globe, from population statistics to economic statistics and so forth.

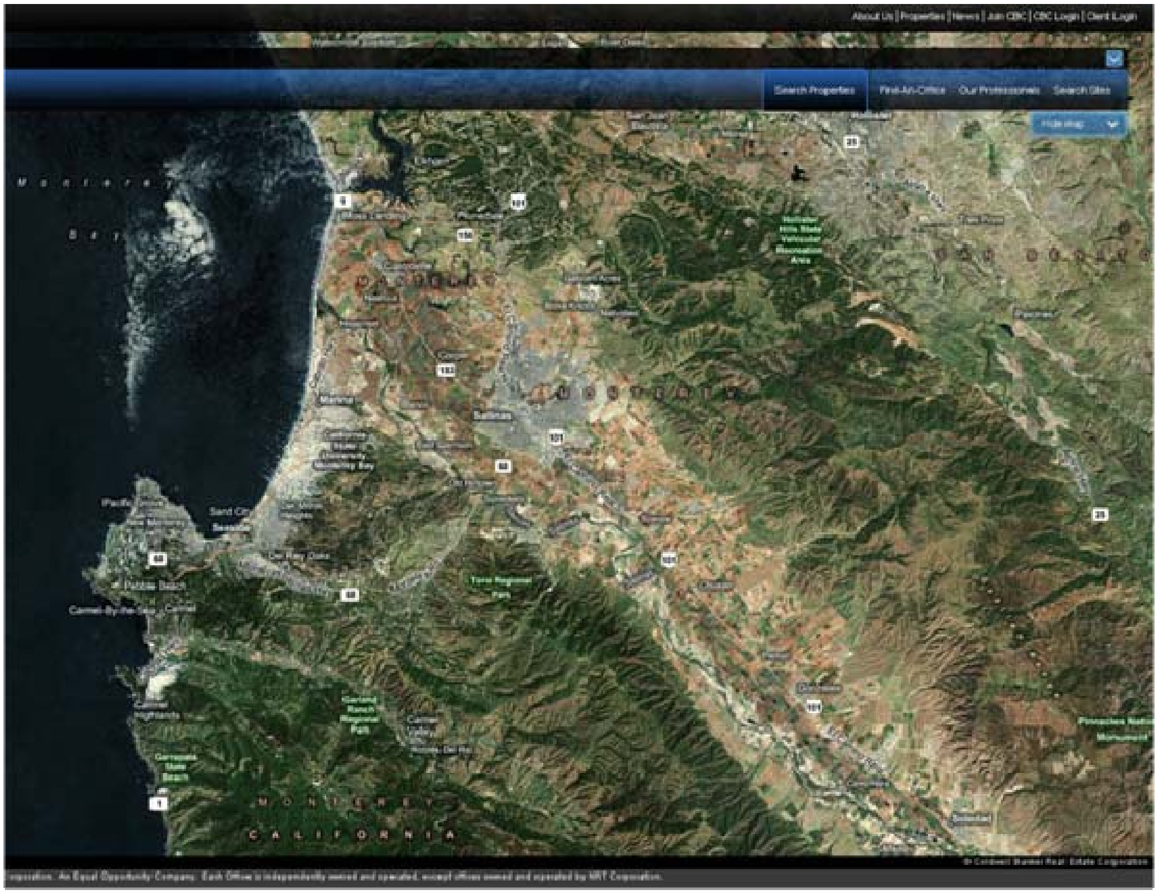

About Us | Properties | News | Join CBC | CBC Login | Client Login
Search Properties
Find-An-Office
Our Professionals
Search Sites
Hide Map
Monterey Bay
Marina
Salinas
Pacific Grove
Sand City
Seaside
Del Rey Oaks
Pebble Beach
Carmel-By-the-Sea
Carmel
Toro Regional Park
Garland Ranch Regional Park
Carmel Valley
Pinnacles National Monument
MONTEREY
CALIFORNIA
© Coldwell Banker Real Estate Corporation
Corporation. An Equal Opportunity Company. Each Office is independently owned and operated, except offices owned and operated by NRT Corporation.

Figure 7. Screenshot of Bing Maps Platform on Coldwell Banker Commercial Web Site

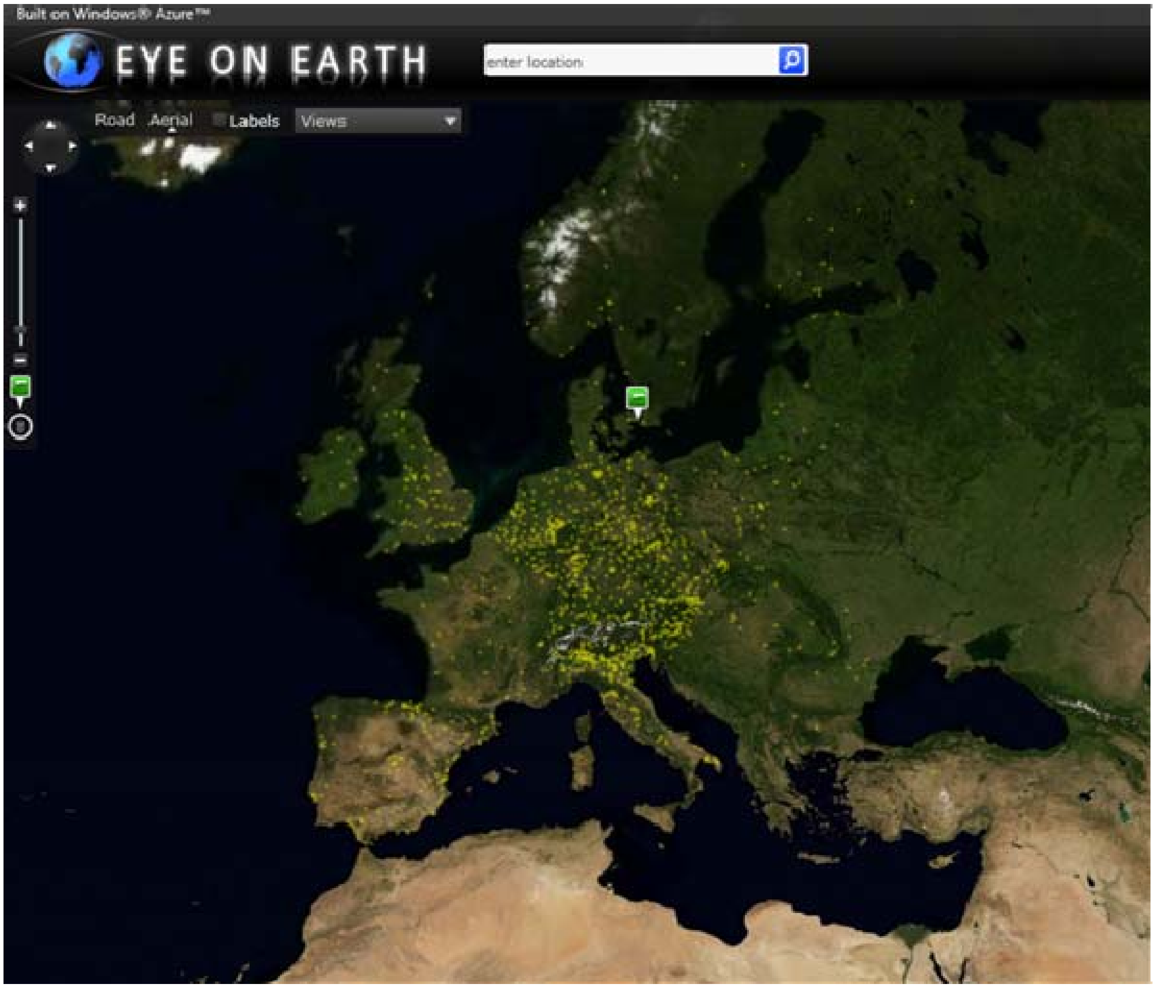


Figure 8. Screenshot of Bing Maps Platform From Eye on Earth Web Site

2. Source Data

The sources of data for Bing Maps Platform are Streetside and Bird's Eye View. These two images are commercial images. The Streetside is an imagery set that allows users to zoom down to the street level look at the area's features (BING, 2010a). Bird's Eye View is an imagery set that allows the user an aerial view from 45 degrees. This imagery only covers the major metropolitan cities of the United States, Canada, Western Europe and Australia (BING, 2010a). Further information about Bing Maps Platform can be found at http://www.microsoft.com/maps.

3. License

The full license agreement for BING Maps Platform can be found at http://www.microsoft.com/maps/product/terms.html. There are multiple options for purchasing and using BING Maps Platform, from Educational use to Developers. For Developers the price is based on a "Limit of 125,000 sessions or 500,000 transactions in a 12 month period" (BING, 2010b). There is no price listed for this type of license. For the Enterprise license, again no prices are listed, but it has three types of pricing models. They are based on usage, number of listed users, and number of tracked assets. The government license has similar pricing models as the Enterprise license, though possibly at a different rate. The Broadcast and Education licenses have no information on pricing models. Finally, the Not-for-Profit license is completely free. All of these licenses have the same amount of access to data.

E. VTERRAIN

VTerrain, Virtual Terrain Project, or VTP, is a project that collects terrain, imagery, and 3D models in one location. "The goal of VTP is to foster the creation of tools for easily constructing any part of the real world in interactive, 3D digital form" (VTP, 2010). The project Web site has numerous links that have descriptions of numerous sources of data and external links that lead you to the data source Web sites. These links and descriptions are broken into the following groups: Elevation, Ground Detail, Rendering, Data Sources and Formats, Culture, Plants, The VTP Software, Other Terrain Software. The project is a great source for finding data sources and other projects that are currently being worked and those that are complete. More information can be found at http://www.vterrain.org.

F. OPEN GEOSPATIAL CONSORTIUM (OGC)

The best explanation of OGC is from a White Paper titled "The OGC – A Unique Organization Offering Unique Benefits":

> The OGC is a not for profit, international voluntary consensus standards organization. The core mission of the OGC is to develop standards that enable interoperability and seamless integration of spatial information,

> processing software, and spatial services. Spatial information and processing encompass geographic information systems (GIS), remote sensing, surveying and mapping, navigation, location based services, access to spatial databases, sensor webs, and other spatial technologies and information sources (Bacharach, 2004).

1. Standards

OGC offers users, researchers, and developers an avenue for changing the overall direction of geospatial technology. The OpenGIS standards developed and continually modified by OGC offers a number of benefits for all of the above. Leverage existing investments in legacy content and applications.

2. Services

OGC also provides Web Services, collectively referred to as OWS. These Web Services are "an evolutionary, standards-based framework that enable seamless integration of a variety of online geoprocessing and location services" (Doyle & Reed, 2001). OWS allows multiple geoprocessing tools to interoperate and communicate across the Web using current Web-based technologies. OWS is an open framework that remains independent of any commercial software and provides a means to link multiple programs and data types together in geoprocessing tools (Doyle & Reed, 2001). Figure 9 illustrates the generic framework of OWS.

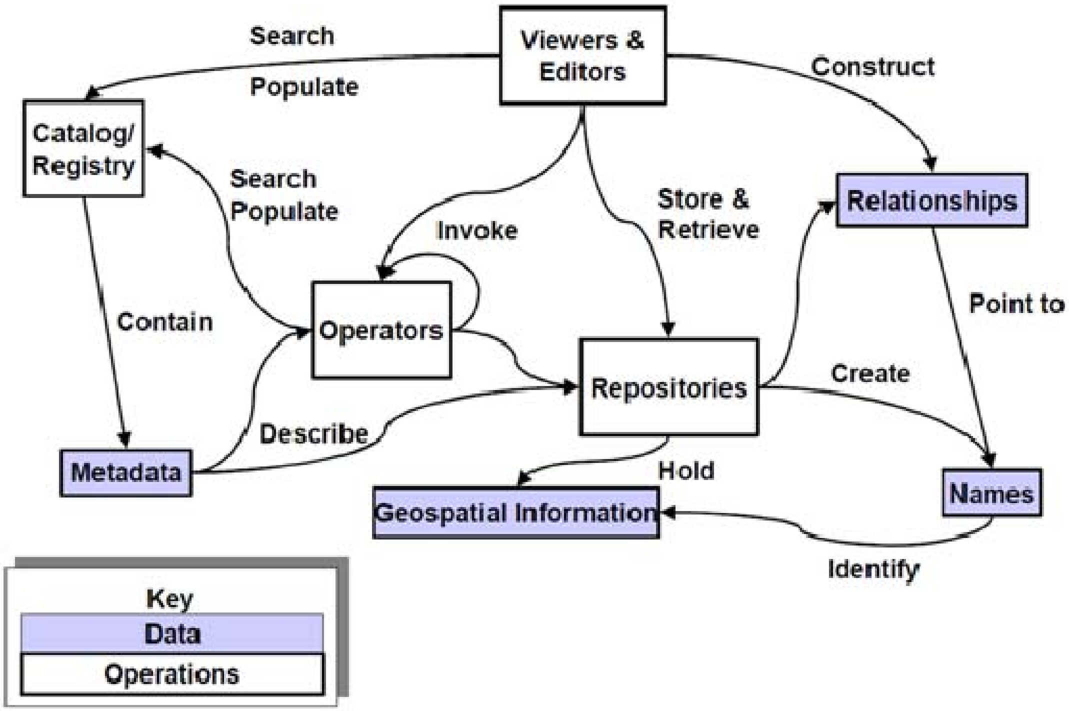


Figure 9. Generic Architecture of OGC's Web Services (From Doyle & Reed, 2001)

Further information about OGC can be found at its home page, http://www.opengeospatial.org/.

G. WEB3D CONSORTIUM

"The Web3D Consortium is a non-profit, international standards organization that has spearheaded the development of the VRML and X3D specifications" (Web3D, n.d.a). Web3D Consortium is currently developing the X3D specification, for 3D graphics to be displayed on the Web. The process of expanding these specifications and standards is ongoing.

1. Mission

The following is the mission statement of Web3D Consortium (n.d.a):

> The Web3D Consortium will support a collaborative environment and drive programs to develop and advance an open standards, royalty free 3D interchange format based on XML, along with tools to represent and

communicate 3D scenes and objects between diverse authoring and presentation hardware and software on the Web, distributed networks and mobile devices.

2. Goals

Web3D has two overall goals. First is the goal to increase the interoperability of all simulation applicable applications which is achieved by reducing the industry fragmentation. Second, the goals are to ensure that the data used in these simulation applications are long lasting and fully interoperable (Web3D, n.d.a).

3. Working Groups and Their Objectives

This section needs a simple table containing the Working Groups and their objectives.

Group	Objectives
X3D	To "make the native authoring and use of declarative XML-based X3D scenes as natural and well-supported for HTML5 authors as the support provided for Scalable Vector Graphics (SVG) and Mathematical Markup Language (MathML)" (Web3D, 2010).
X3D-Earth	Open-source and Open-standard Geospatial Model representation on the Web.
CAD	"Increasing the value and cross-application use of CAD data through X3D" (Web3D, n.d.b)
H-Anim	Developing a standard for modeling human animation on the Web
Medical	Provide a open-source, Web-based solution for viewing medical information without the need for medical specific workstations

Figure 10. Web3D Working Groups

H. SUMMARY

The above chapter covers the related work for this thesis. It describes Google Earth and its capabilities, data sources and licensing. It describes NASA World Wind and its capabilities, data sources and licensing. It describes BING Maps and its capabilities, data sources and licensing. The chapter also describes working groups and consortiums that continue the work of geospatial models, i.e., OGC and Web3D.

III. PRIOR WORK: DESIGNING X3D-EARTH GLOBE CAPABILITIES

A. INTRODUCTION

This section provides an introduction to the design process of the X3D-Earth capabilities and history of X3D-Earth efforts.

B. GREG LEAVER THESIS: INITIAL EFFORTS

This section covers the initial efforts of Greg Leaver's thesis for X3D-Earth (Leaver, 1998).

1. Background

Greg Leaver had an interest in using data from the Monterey Bay National Marine Sanctuary (MBNMS) in generating a 3D model. The problem of his thesis was the need to create an easy method for generating a 3D scene from terrain data for use in scientific research and visualization. His use of the MBNMS terrain data was based on the accessibility to the data via partnerships between NPS and the Monterey Bay Aquarium Research Institute (MBARI).

Leaver used the 3D modeling language VRML to represent the 3D terrain. Within VRML, there is an *ElevationGrid* node, which represents the terrain in 3D. His thesis work was to utilize this node in creating the 3D environment. In the time of his thesis, there were a number of commercial tools for converting terrain data to VRML, but Leaver intended to create an open source generator to avoid the costs and limitations of using a proprietary tool (Leaver, 1998). VRML is the direct predecessor of X3D and so this work remains directly relevant.

2. Creating the Globe in VRML

The bathymetry from the MBNMS was converted into a gridded terrain dataset using code written by Ray McClain (Leaver, 1998). This dataset was broken into three levels of detail based on the resolution of the gridded datasets: 2000-meter, 1000-meter and 500-meter postings (Leaver, 1998).

Leaver created two Java programs to produce two types of VRML files to represent the MBNMS is 3D (Leaver, 1998). One was written to convert the terrain to VRML terrain syntax and the other was to create the VRML scene trees, which provide the basis for the level of detail based on the viewer's position (Leaver, 1998).

The first program, CreateVrmlTile, reads the data file name, the metadata, the Elevation data and generates the VRML syntax for storing the data in a VRML scene. This tile is geographically referenced based on the files name and metadata. These files are then ready for use by the second program.

The second program, CreateVrmlTree, takes the previously generated files for all levels of detail and generates a scene that links them together. Each "tree consists of one parent and four children," and this is repeated for all levels, creating a complete chain of connections from the lowest resolution to the highest resolution data (Leaver, 1998).

3. Results

Leaver produced a 3D model for the MBNMS that was viewable from any computer with Internet connection and a browser that was VRML compatible. The scenes were relatively slow to load and render using the hardware and software of that era, but his work proved that this type of format for designing and rendering large 3D terrain scenes was possible. His work laid the foundation of future work in this field.

4. Recommendations

The thesis includes a number of recommendations for advances in browser implementation and VRML node modifications. The most important recommendation is to add more levels of resolution to increase the speed of rendering. He also recommended that using the original data, which was in much higher resolution, 10 meters, to generate higher resolution 3D scenes, would be good future work. With today's processor speeds and computational power, this recommended approach is now possible.

C. GEOVRML

1. What is GeoVRML?

"GeoVRML is an official Working Group of the Web3D Consortium. It was formed on 27 Feb 1998 with the goal of developing tools and recommended practice for the representation of geographical data using the Virtual Reality Modeling Language (VRML)" (GeoVRML.org, 2002). The goals of the group are to create a language that can enable geospatial data to be viewed the any Web browser that has a VRML plug-in.

The latest version is GeoVRML 2.0, which is the direct predecessor to X3D-Earth.

2. Goals of GeoVRML

The ultimate goal of GeoVRML is to provide a standard for representing geospatial data in 3D over the Internet. Using VRML as a building block to create the GeoVRML language is a critical step for achieving this goal. Because it is single precision, VRML is inaccurate and too general, yet it serves as a good starting point. The group intended to decipher what geospatial information was important for creating the model (GeoVRML.org, 2002).

3. Issues

a. Coordinate Systems

There are dozens of coordinate systems for mapping the globe. GeoVRML has code for translating these coordinate systems and has a wide array of coordinate systems for use in modeling. With VRML as a basis, there was not enough memory to adequately represent the coordinate system based on the Cartesian coordinates of VRML. Therefore, double precision was needed. Which coordinate system would be the system for GeoVRML? The initial intent was to have twelve coordinate systems in GeoVRML 1.0, but a number of other coordinate systems have been added for Version 2.0 (GeoVRML.org, 2002).

b. Time Referencing

In version 1.0, there is no time referencing. The proposal is to use Coordinated Universal Time (UTC) in future versions (GeoVRML.org, 2002).

4. Continuing Work

Mike McCann, researcher with Monterey Bay Aquarium Research Institute, continually uses GeoVRML in modeling research dives for Remote Operated Vehicles. Mr. McCann is a leader in geospatial 3D modeling over the Internet. He is currently co-chair of the X3D-Earth Working Group, with Professor Don Brutzman.

D. HITTNER THESIS: SCENE GRAPH DESIGNER FOR GEOSPATIAL REPRESENTATIONS

1. Background

Brian Hittner's thesis describes a method for "rendering large-scale terrain models in 3D" (Hittner, 2003). The background for his thesis is contemporary work in 3D models, X3D, GeoVRML, and the use of DTED terrain files.

Hittner describes the various geospatial datums commonly used in 3D terrain modeling. The most common is World Geodetic System 1984, (WGS 84). This datum is used in navigation and charting for the U.S. Navy and many foreign navies.
WGS 84 is also used in the DTED and SRTM terrain representations, along with a number of other terrain files. The other commonly used coordinate system is Universal Transverse Mercator (UTM), which represents location based on meters north and east from an origin, set at 180 degrees and the equator (Hittner, 2003).

Hittner describes the open-source 3D modeling software that was used in his thesis to produce X3D and VRML. At the time of his thesis, X3D was in its infancy and had limited browser support. VRML was fully supported in a number of browsers and there were XSLT stylesheets available to convert the X3D files to VRML. Thus, he used these to convert his X3D files to VRML for rendering. He describes the capabilities of X3D for rendering and positioning complex objects, as well as the use of GeoVRML to model 3D terrain. Figure 11 shows an example of the capabilities of X3D in modeling complex vehicles. Figure 12 shows an example of the use of GeoVRML for rendering 3D terrain.

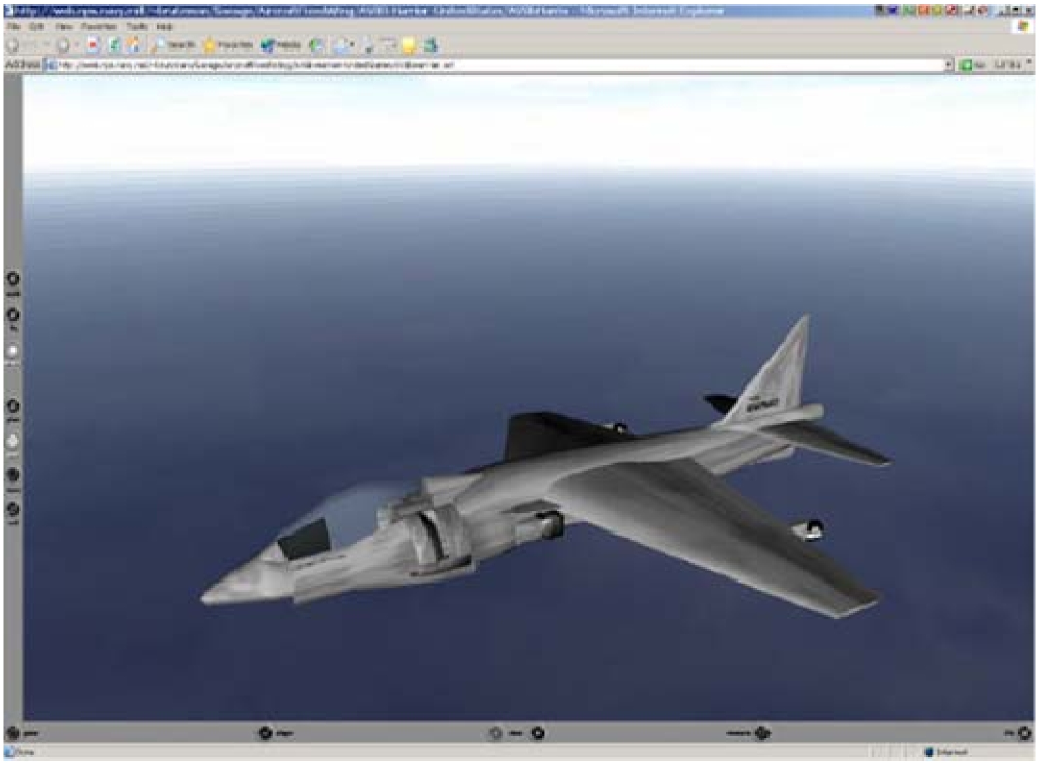


Figure 11. AV-8B Harrier Modeled by Miguel Ayaya (From Hittner, 2003)

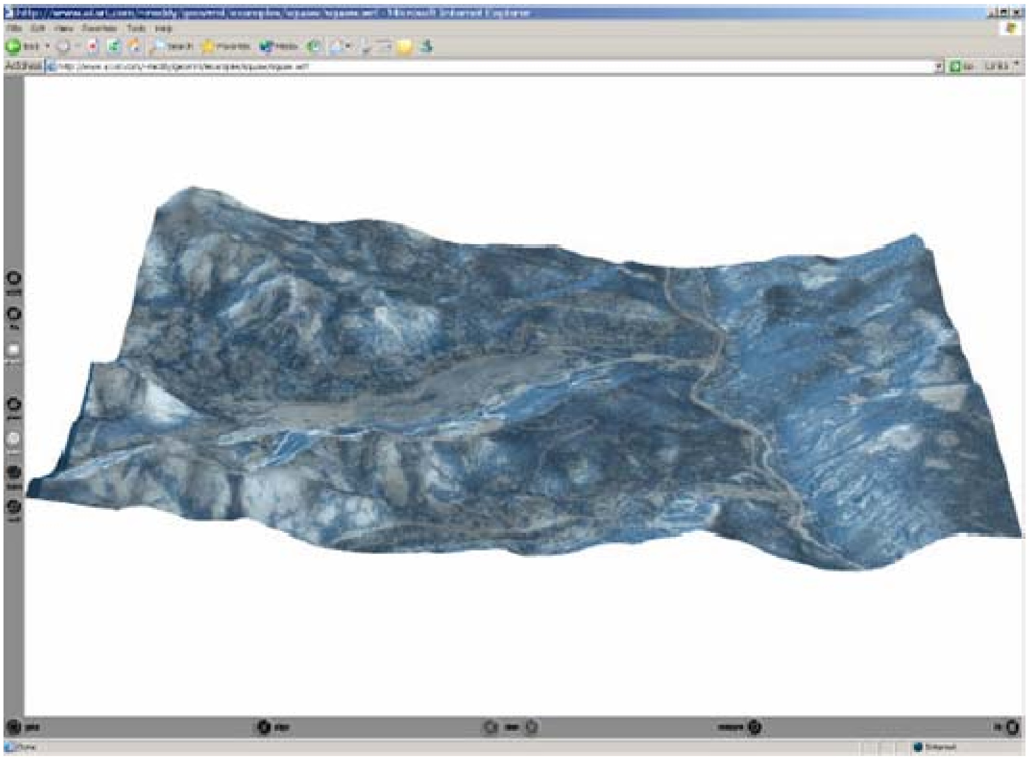


Figure 12. Squaw Valley Modeled by Martin Reddy (From Hittner, 2003)

2. Creating 3D Terrain

Within his thesis process, Brian Hittner had two major objectives. The first was to develop an automated method for rendering terrain in 3D. The second was positioning and orienting 3D objects with respect to his generated 3D terrain.

The first task he needed to complete with respect to rendering terrain in 3D was to develop a method of generating the terrain from existing height arrays. These height arrays were derived from existing terrain files; DTED terrain files were used in his thesis. These terrain files were converted to height arrays, which are ordered values that represent the height of the terrain at evenly spaced locations. These heights were then converted into 3D terrain using the *GeoTerrainGrid* node within GeoVRML. To complete this, Hittner created a Java program, GeoManager, for this task. The GeoManager class reads in the height array and creates these elevation grids for use in the VRML file. Figure 13 shows an example of the *GeoTerrainGrid* node.

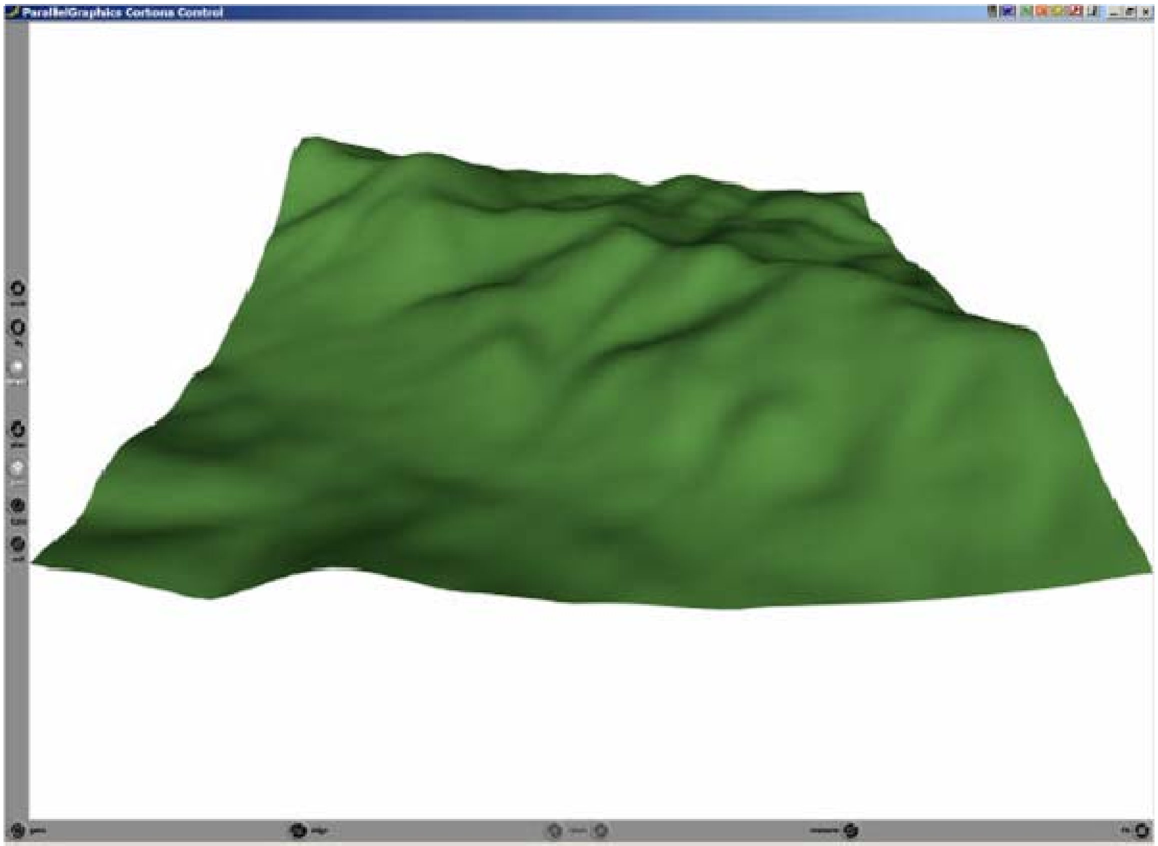


Figure 13. *GeoTerrainGrid* Example (From Hittner, 2003)

The GeoManager class can be used for multiple 3D terrain tiles within a specific scene. Therefore, multiple tiles can be rendered in unison to render a larger area and manage the terrain data for all the tiles.

The next major objective for Hittner's thesis is the positioning and orienting of objects with respect to the 3D terrain. This task was accomplished by the use of a new node created by Hittner in his thesis process. This node is the *GeoLocation3* node. VRML did not have support for positioning and orienting objects based on elevation data. Therefore, Hittner developed methods for achieving this within the *GeoLocation3* node. This node allows a user to position and orient any 3D object automatically when using the node. There are two attributes for the node, *autoSurfaceOrientation* and *autoElevation*. These two attributes can be set to true or false, which enables or disables, respectively, the methods that position and orient the object with respect to the elevation data. The *autoSurfaceOrientation* attribute is the mechanism for orienting the object based on the slope of the terrain. The *autoElevation* attribute determines the height of the object and, if set to *true* it is placed on the surface of the terrain. For example, if representing a vehicle, such as a tank, both attributes are set to true, which places the object in contact with the terrain and tilts it based on the slope (Hittner, 2003). An example of both attributes being false would be a helicopter or other aircraft in which there is no need for the object to be in contact with the terrain or tilted with respect to the slope (Hittner, 2003).

3. Recommendations

Hittner has a number of recommendations for future work. The one that is most related to this thesis is the use of Level of Detail to minimize the number of polygons rendered in the 3D terrain scenes. He proposed having this capability be organic to the GeoTerrainGrid node or a similar node. This would enable the node to determine which areas of a grid needed to be rendered at higher or lower resolution. This might be an opportunity to optimize a smaller number of overall files to cover a specific region. Yet this approach might also mean that all the X3D or VRML files for a given DTED terrain file have very large sizes. Any corresponding high-resolution imagery that is also used becomes very large as well, slowing down the rendering process and perhaps overloading graphics-card memory capacity. A similar method would be needed to vary the level of detail for the imagery. An alternative to this method is the generation of multiple files to represent the levels of detail, the method used in this thesis.

E. X3D-EARTH TECHNICAL REQUIREMENTS WORKSHOP

1. X3D-Earth Goals

This section describes the goals of X3D-Earth as described in the workshop.

“Web3D Consortium members are preparing to build a standards-based X3D-Earth usable by governments, industry, scientists, academia and the general public” (Brutzman et al., 2007). X3D-Earth is intended to be used to map terrain, cartography, and imagery which is available to be used with X3D model. This uses open standards, XML and Web architectures to create an extensible globe model that can be viewed from any X3D-enabled browser.

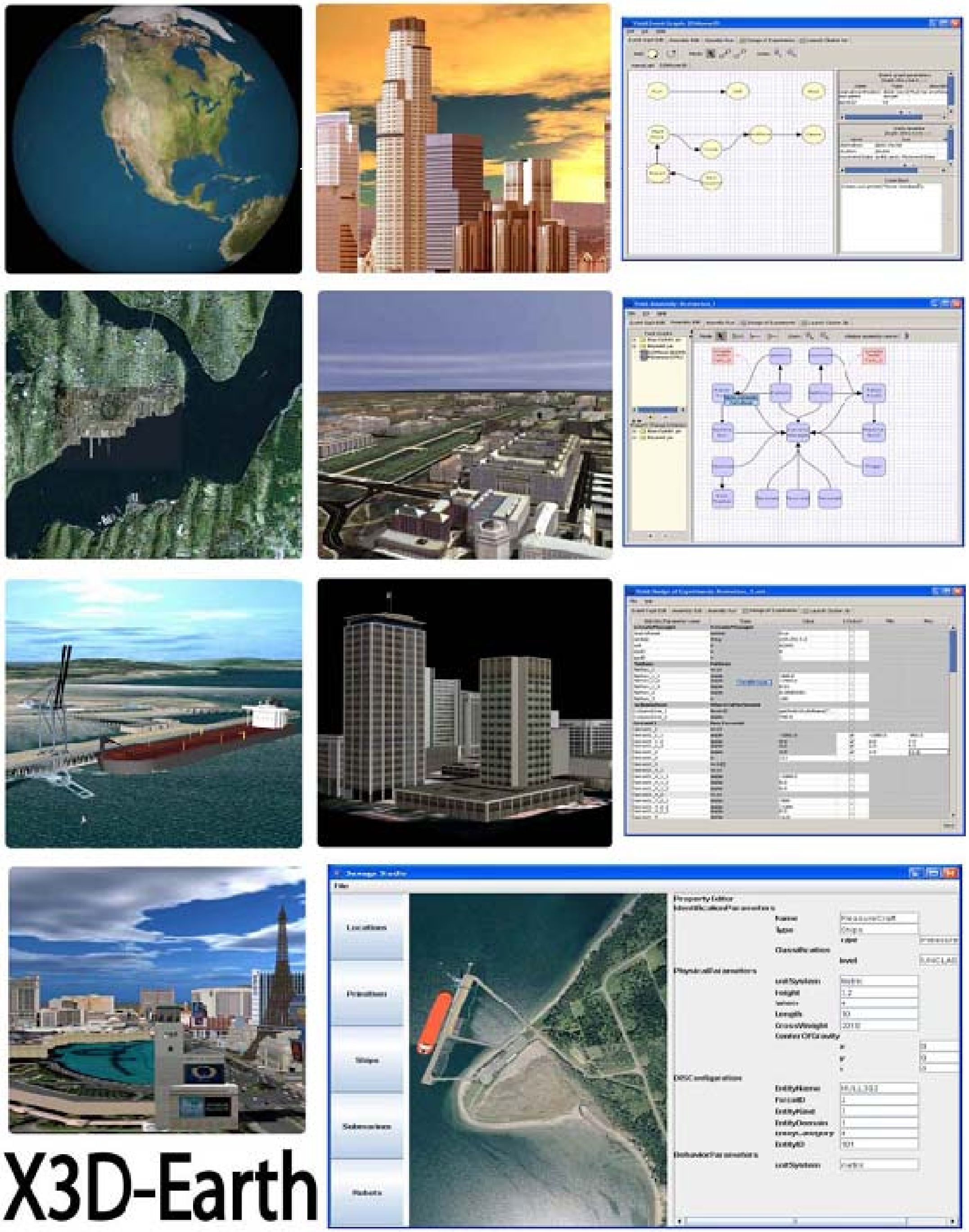


Figure 14. Montage of Visual Concepts Courtesy of Aniviza, NPS MOVES, Planet 9 Studios, and Yumetech Inc. (From Brutzman et al., 2007).

2. Technical Requirements

The following section is taken from the X3D-Earth Requirments Workshop, and a white paper on the proposed requirements from Alan Hudson (Brutzman et al., 2007).

The X3D-Earth initiative should:

- Provide support for viewing the Earth from space to the surface of the globe,
    - To include subsurface features like mines, changing of the terrain to support construction sites, and bathymetry
- Require each participating organization or user to contribute computing resources, i.e., bandwidth, storage and processing
- Have the following server requirements:
    - Provide a reference Server Architecture
    - Have at least one Open Source implementation
    - Provide multiple versions of X3D-Earth for supporting local data sources and other potentially classified materials
- Have the following client requirements:
    - Have at least one Open Source implementation
    - Easy to navigate, i.e., from space, from the surface and from the subsurface.
- Provide a way for distributing the world state of the model
- Provide a way to distribute the chat areas in the world model
- Display volume data with respect to the terrain data, i.e., cloud and weather patterns on the terrain.
- Provide a community object authoring capability, i.e., 3D buildings, or GIS information that can be sorted and added to any other scene
- Enable client implementers to differentiate which is best, i.e., which renders the best based on architecture
- Enable multiple planets to be rendered and navigated
- Provide user selected viewing mode of raw data or derived models
- Provide for data fusion of the multiple data sets with other features, such as KML push pins and routes
- Provide the ability to represent and render the interior of buildings

A more detailed description of these requirements and the e-mail discussions of the requirements can be found in the X3D-Earth Technical Requirements Workshop Report at http://www.web3d.org/x3d-earth/workshop2006/report.html.

3. Conclusions and Recommendations

This section describes the conclusions and recommendation of the workshop.

There were numerous arguments for the need for X3D-Earth. The conclusions were broken into what the community was seeing, not seeing, observations and the workshop conclusions.

They saw many success stories and doable technical approaches. There were standards and standards organizations that were available for work in the X3D-Earth process. There were many diverse projects (Brutzman et al., 2007).

They were not seeing "coherent use cases for design requirements" (Brutzman et al., 2007). There were no major controversies or conflicts over the requirements for X3D-Earth. Nor was there another 3D format that would be able to do the same as X3D-Earth. Nor is there "confusion about what is needed next" (Brutzman et al., 2007). But there is no agreement on the server architecture and content that is needed in X3D-Earth.

They observed that commercial products had the best quality, yet open source versions could compete. There were a number of open source and freely available data sets for use in these models (Brutzman et al., 2007).

The workshop conclusions state that X3D-Earh is a feasible option and there are many resources and projects available to begin the process. They did not find anything to hold up the process and there were a number of use cases to start from. Therefore, the process of starting the initiative was undertaken shortly after the workshop.

F. THORNE DISSERTATION: RESOLVING GEOORIGIN-BASED FLOATING-POINT ROUNDOFF PROBLEMS

This section discusses Dr. Chris Thorne's dissertation and its relevance to this thesis. Dr. Thorne's program, Rez, has also had a direct impact on this thesis. This program was created in conjunction with his PhD dissertation work.

1. The Problem of Jitter in 3D Globes

Prior to his dissertation work, there were issues with representing a full 3D globe with single-precision floating-point numbers. Such values appear to provide an efficient way of representing distances from an origin in XYZ coordinates for nearly any 3D simulation. The coordinates of the various objects in the simulation were relatively small in file size. Yet, with this method of representing coordinate points in geographic reference to the origin of the globe, it becomes a serious problem. In representing the origin of the globe, the origin is at (0, 0, -6378137). To represent an object at the prime meridian and the equator, i.e., the buoy that marks this location, it would be (0, 0, 0). This buoy alone would have an issue rendering on the simulation. If there is another buoy in a harbor in northern Europe, it is very difficult to represent the coordinates in single-precision floating point. Therefore, the objects "jitter" within the simulation. That is, the objects apparently move a number of meters to the closest coordinate that can be represented in single-precision, which may be meters away. This round-off error is due to the great distance from the origin, which cannot be correctly recorded using single-precision floating-point numbers. Jitter can often be 3–10 meters.

The following figure illustrates an example of the density of floating point numbers. "The step-wise increasing gaps between numbers represents how gap error doubles at each exponent boundary: where the exponent is incremented. Observe that the greatest density of exponent boundaries exists close to the origin" (Thorne, 2007).

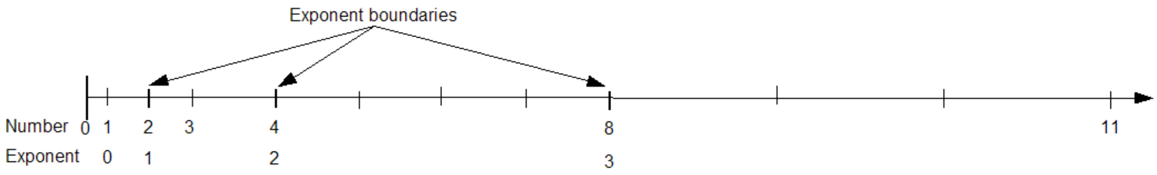


Figure 15. Floating Point Density (From Thorne, 2007)

These gaps only increase as the distance from zero increases, especially for distances greater than the radius of the earth.

2. The Solution: Floating Origin

As a solution, Dr. Thorne proposed creating a "floating origin." This is accomplished by defining an origin for the viewer and every relative object is then translated based on that local origin. Therefore, when viewing the buoy in Northern Europe, the origin would be based on the viewer, not the origin of the earth. Thus the origin would be at most several kilometers away, which is easily represented in single-precision floating point. Thus the "jitters" are eliminated through this combination of position values.

3. Experiments and Results

Dr. Thorne ran multiple experiments to illustrate the errors of rendering multiple objects at great distances from the origin. The following figures illustrate some of his experiments.

In the following series of pictures, there are a number of tiles rendered to the screen. These tiles are numbered from 1 to 54, with one white tile in front of a red tile of the same number. Therefore, at the origin the viewer only sees the white tiles, in perfect order with minimal separation between the tiles. As the model is moved from the origin, tile separation is visible and at great distances from the origin red tiles are rendered. The following pictures are at 70,000 meters, 300,000 meters, 700,000 meters, 2 million meters, 3 million meters, and 5 million meters, respectively.

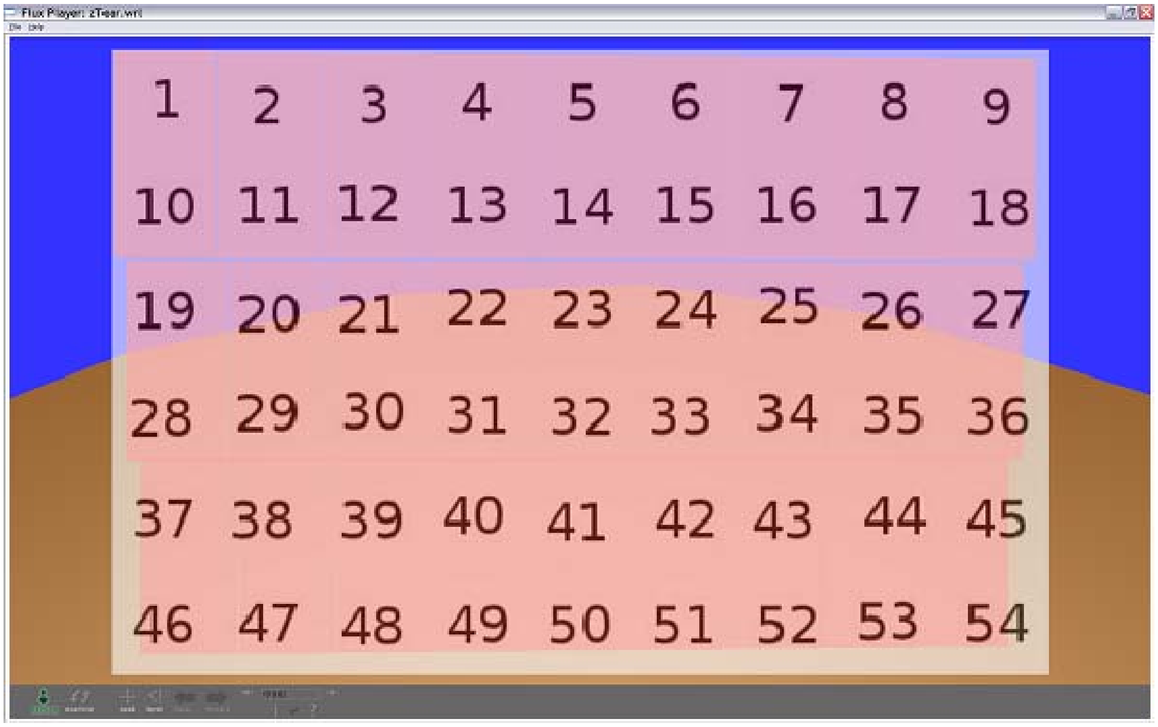


Figure 16. Example Tiling at the Origin (From Thorne, 2007)

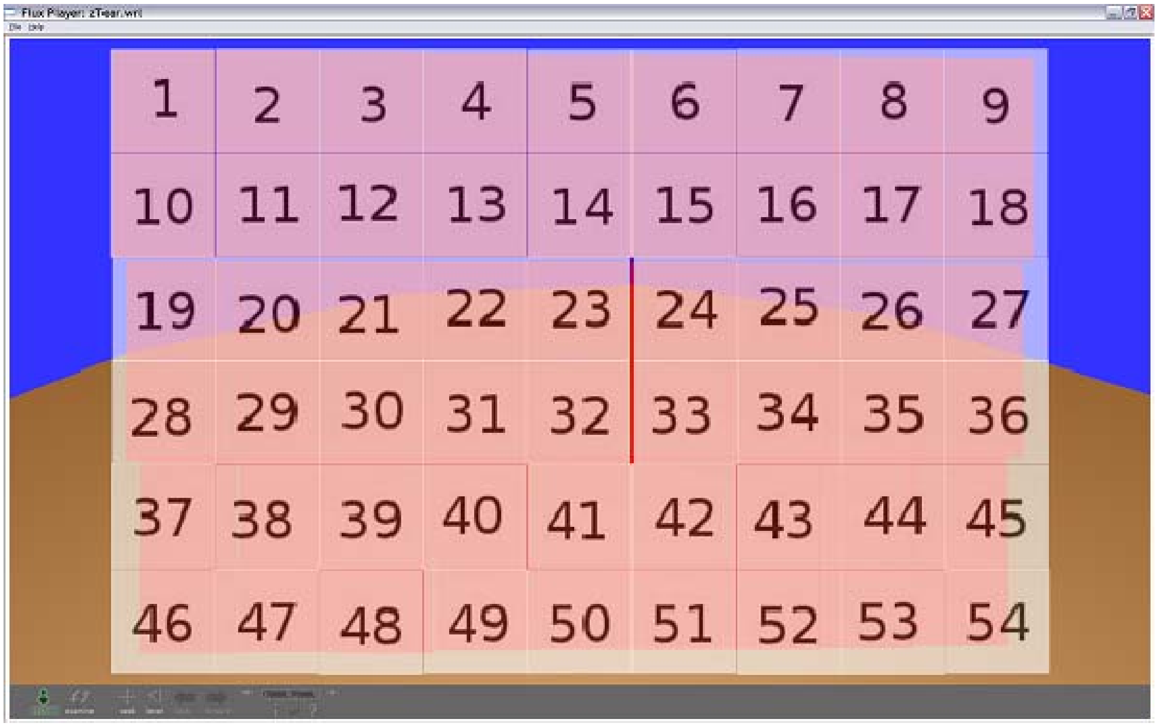


Figure 17. Example Tiling at 70,000 Meters (From Thorne, 2007)

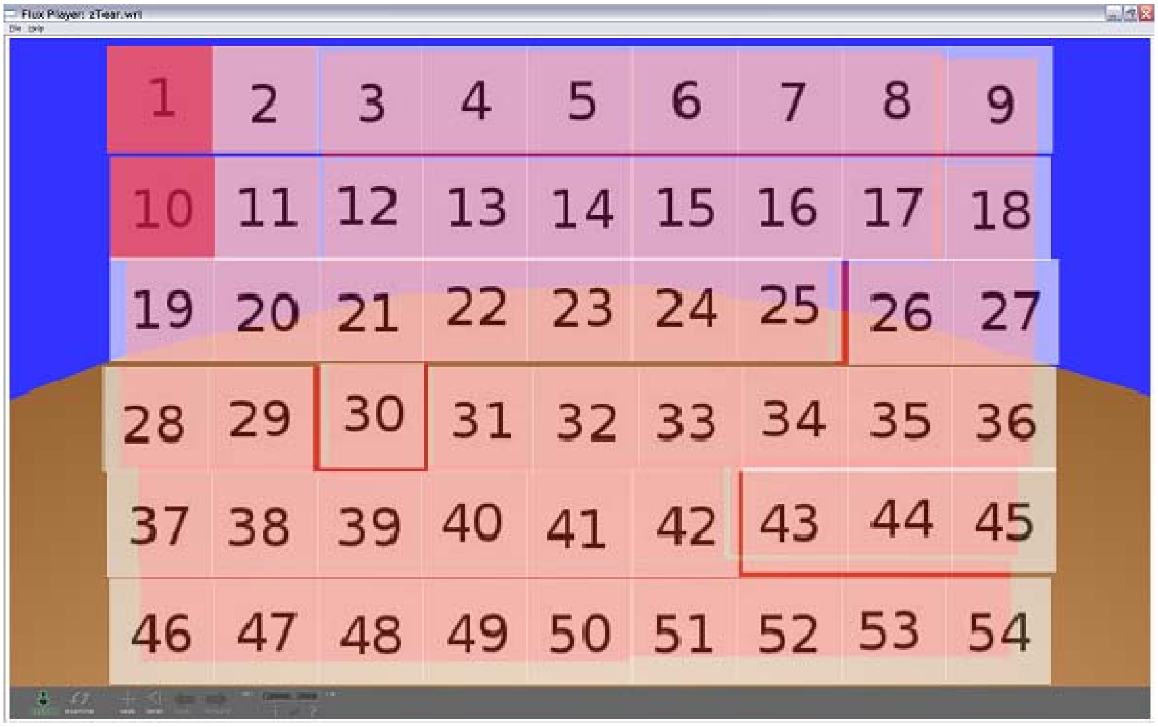


Figure 18. Example Tiling at 300,000 Meters (From Thorne, 2007)

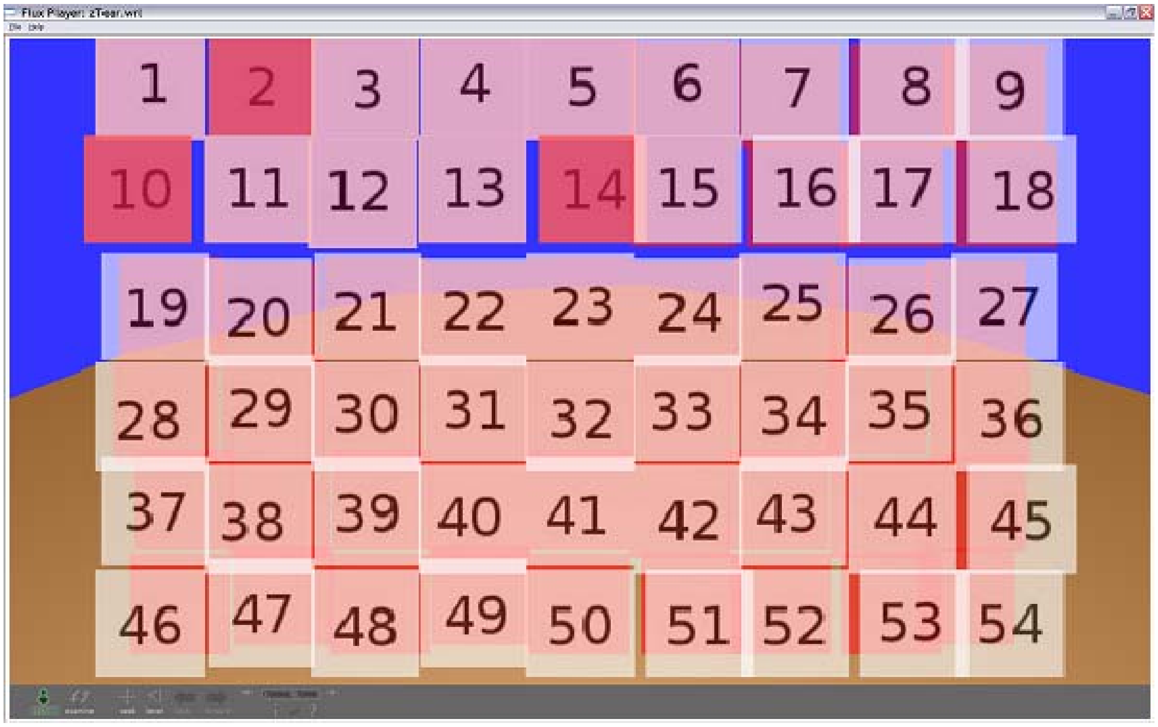


Figure 19. Example Tiling at 700,000 Meters (From Thorne, 2007)

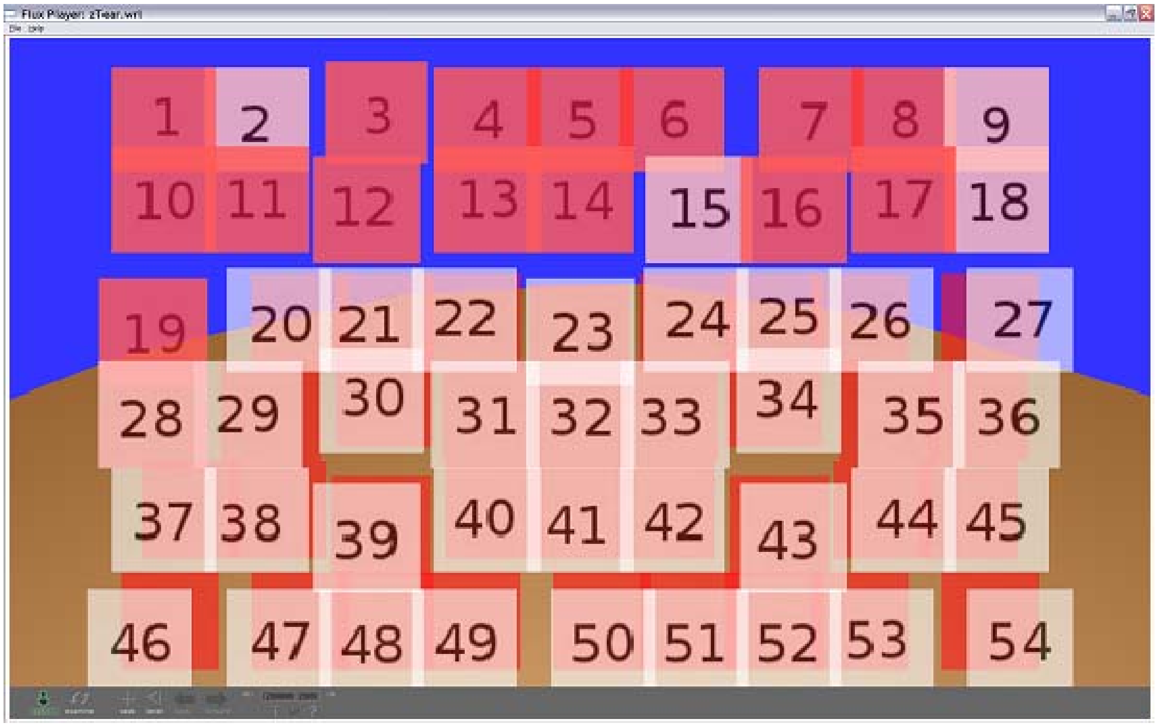


Figure 20. Example Tiling at 2 Million Meters (From Thorne, 2007)

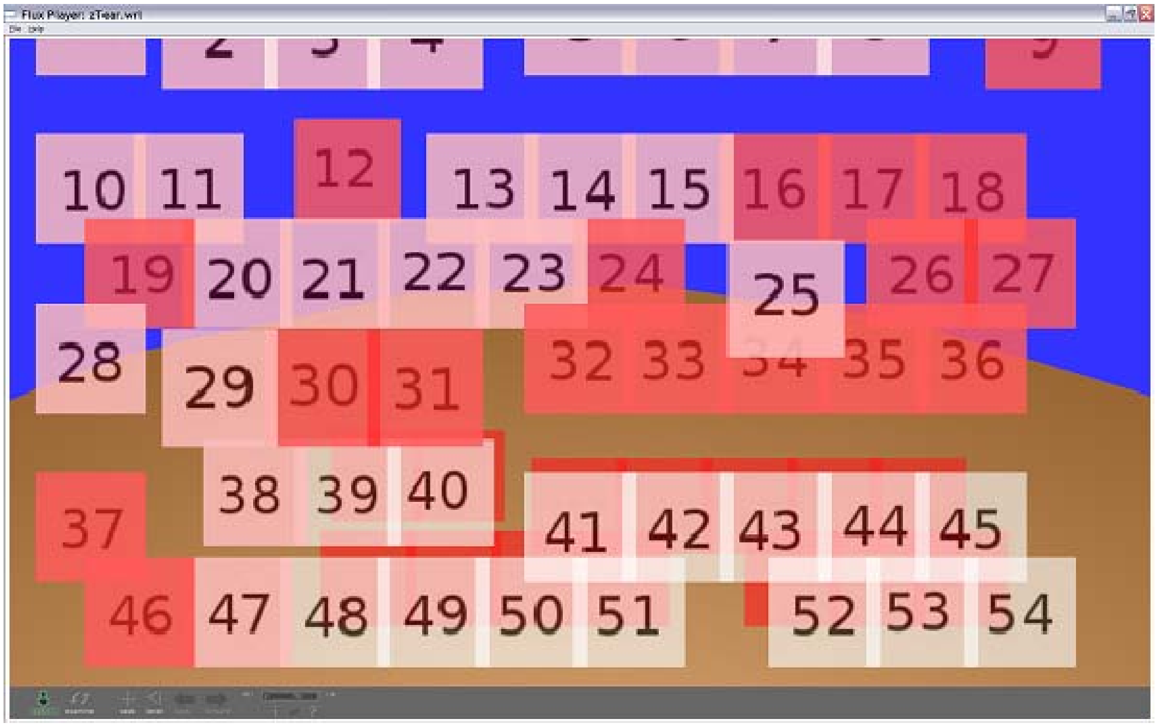


Figure 21. Example Tiling at 3 Million Meters (From Thorne, 2007)

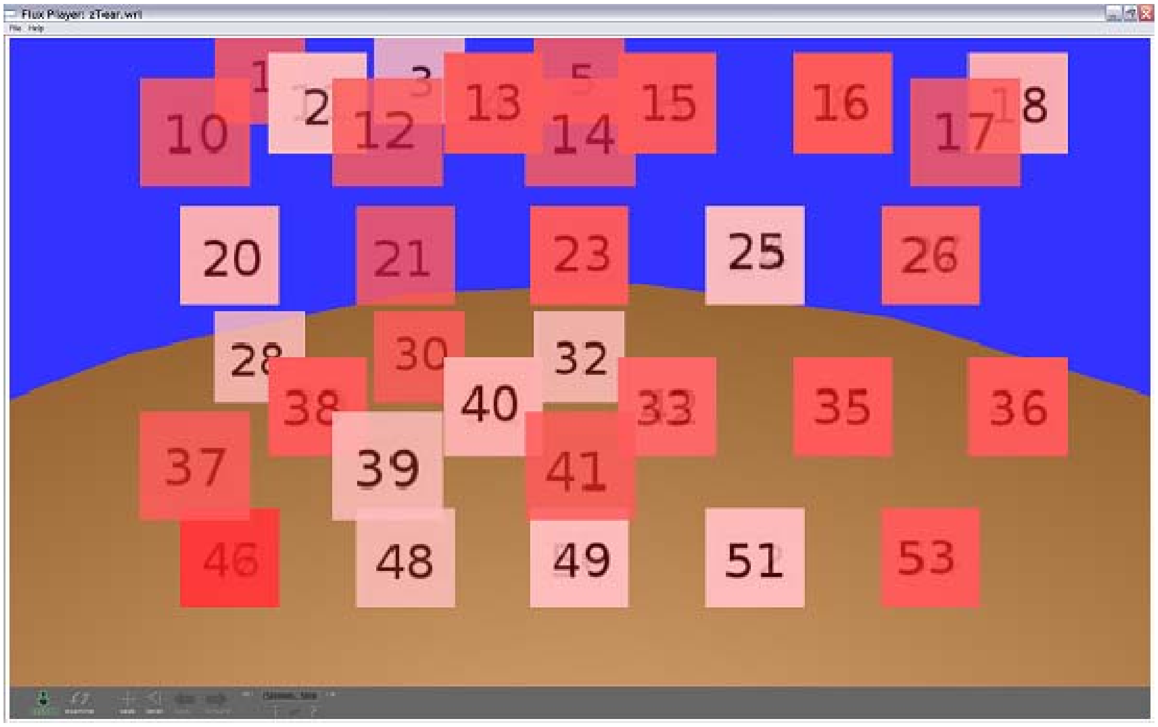


Figure 22. Example Tiling at 5 Million Meters (From Thorne, 2007)

At the maximum distance testing, the jitter can be clearly seen in the results of this experiment. The pink and red tiles are rendered in many of the tiles, as well as the ordering of the tiles is no longer consistent. Note that the value used remains less than the average radius of the earth (6,378,137 meters).

With the floating origin, the picture of the tiles at the origin, instead, matches what the user sees at any location in the geospatial simulation.

G. YOO POSTDOCTORAL WORK: BUILDING REGIONS AND LOW-RESOLUTION GLOBES

Byounghyon Yoo, PhD, was a postdoctoral research associate and Web3D Fellow from 2006 to 2008 at Naval Postgraduate School. While at NPS, he further advanced the work of X3D-Earth. His primary contributions to the effort were his Terrain Tile Production Chain, his proof of concept in generating multiple low-resolution globes, and his OpenStreetMap globe.

1. Terrain Tile Production Chain

The Terrain Tile Production Chain, the following figure, shows the process in which a location is created from freely available terrain data and imagery. “The process generates multi-resolution terrain models in valid X3D format from geospatial information including high-resolution satellite imagery, thus supporting real-time interactive X3D visualization while simulation applications are running” (Yoo & Brutzman, 2009).

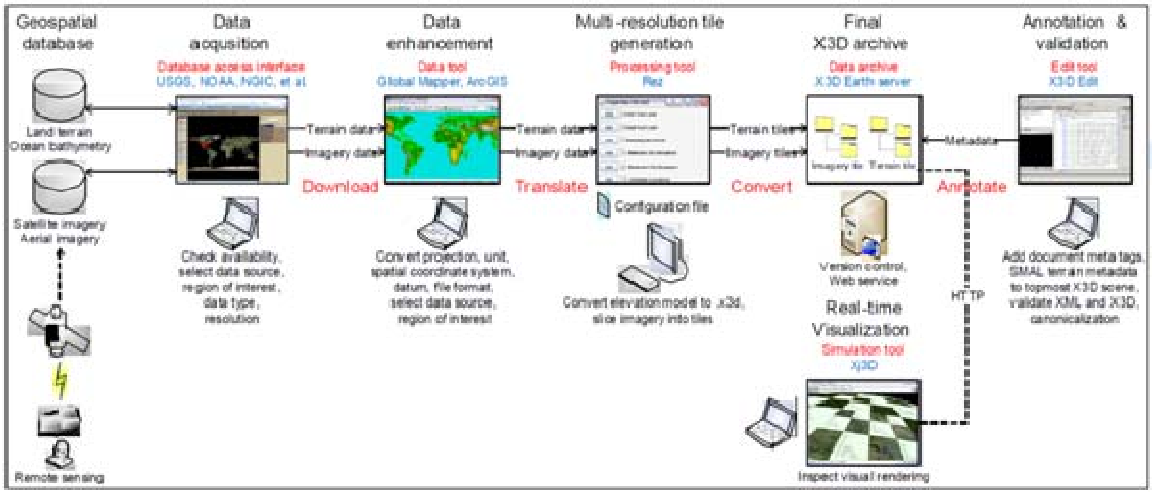


Figure 23. Terrain Tile Production Chain (From Yoo, 2009)

This process consists of five generic steps. The imagery and terrain data is obtained via a local drive or from an Internet source. Following this the data is processed into a file format that Rez can handle. For this production chain, Dr. Yoo used software called Global Mapper. Global Mapper is commercially produced and a proprietary software that has a wide array of geospatial capabilities, specifically viewing and converting geographically referenced imagery and terrain data (more information on Global Mapper can be found at www.globalmapper.com). This software is used to translate the imagery and terrain into a format for use in Rez. The next step is to process the terrain and imagery using Rez via a command line call or a graphical user interface. Rez parses the terrain data and creates the X3D tile pyramids, and separately processes the imagery data into imagery pyramids to match the naming convention of the X3D pyramids. Once these pyramids are created the parent scene is modified via any text editor and used in a number of applications.

This production chain has been utilized at NPS as a student assignment in the Advanced X3D course since its introduction. This method is straightforward, and examples are readily available. The following two figures illustrate the use of multiple levels of detail for San Diego Harbor.

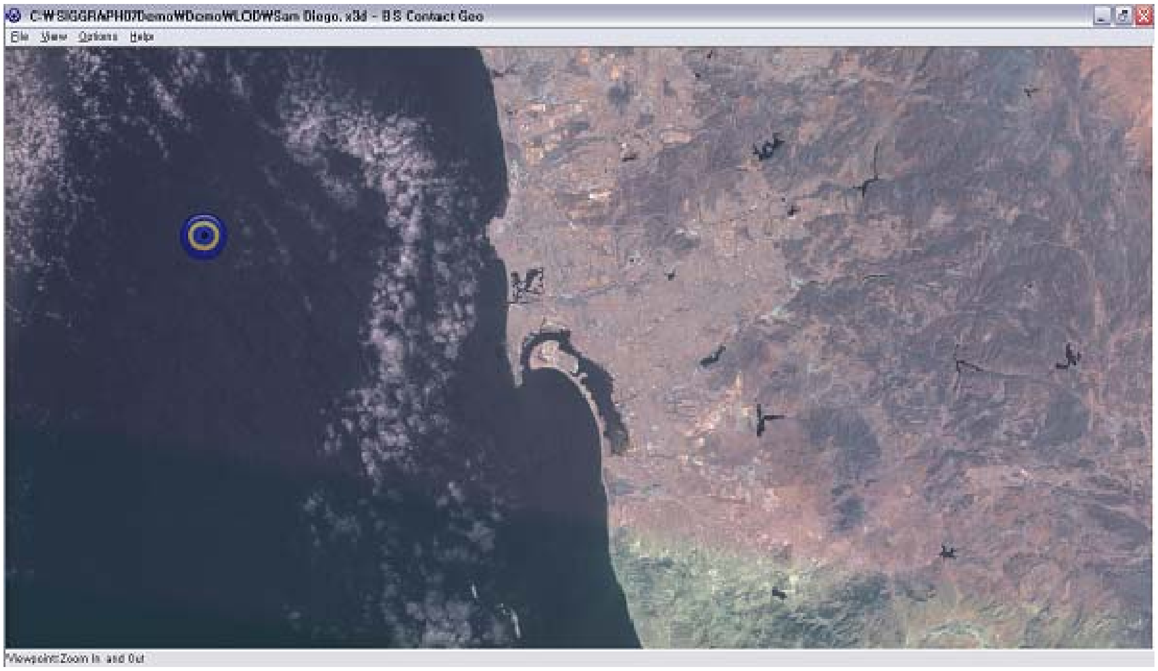

C:₩SIGGRAPH07Demo₩Demo₩LOD₩San Diego.x3d - BS Contact Geo
File
View
Options
Help
Viewpoint:Zoom In and Out

Figure 24. San Diego, California, Low Level of Detail (From Yoo & Brutzman, 2009)

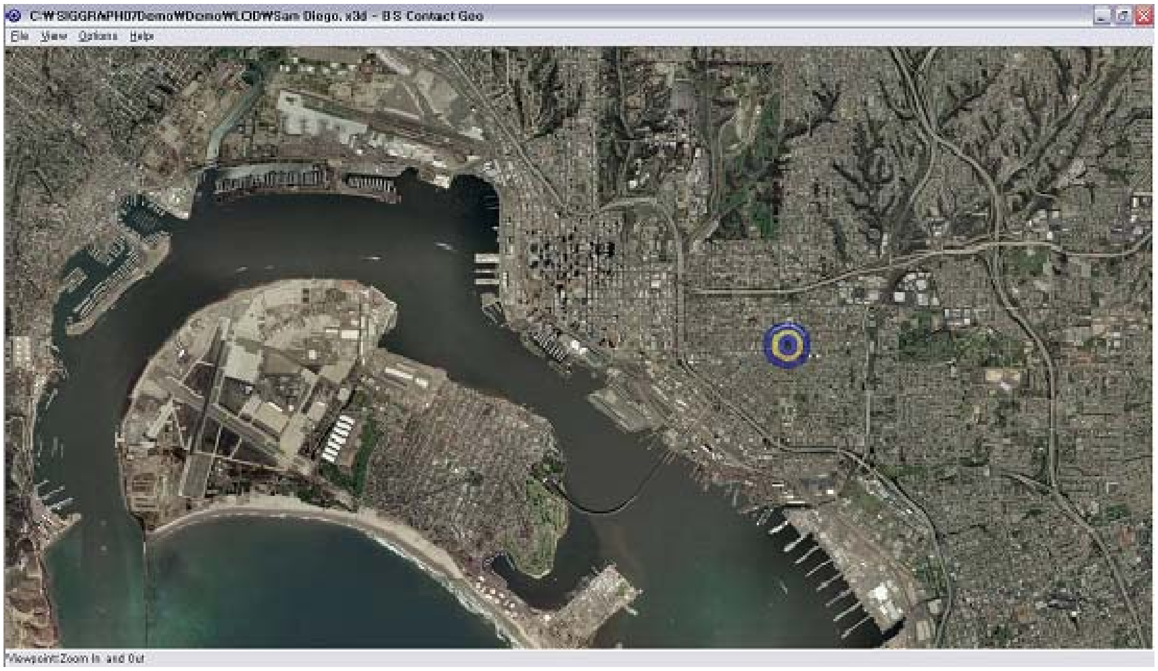
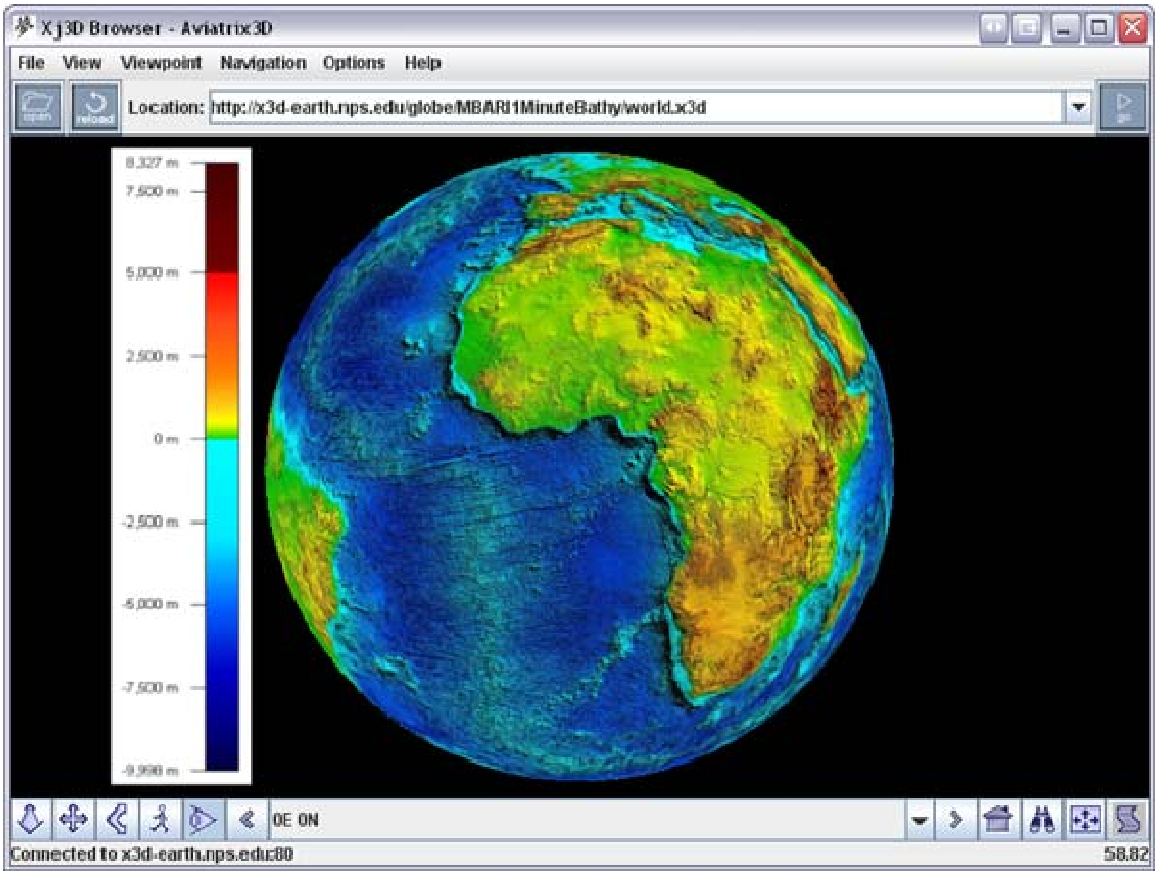

C:₩SIGGRAPH07Demo₩Demo₩LOD₩San Diego. x3d - B S Contact Geo
File View Options Help
Viewpoint:Zoom In and Out

Figure 25. San Diego, California, Higher Level of Detail (From Yoo & Brutzman, 2009)

2. Low-Resolution Globe Generation

Another major contribution to the X3D-Earth effort was the generation of a number of low-resolution globes. Utilizing the Terrain Tile Production Chain, Dr. Yoo was able to produce low-resolution globes from freely available imagery and terrain data. For the example illustrated in Figure 26, Dr. Yoo produced a globe from 1-Minute resolution bathymetry data obtained from the Monterey Bay Aquarium Research Institute (MBARI). This data consists of bathymetric soundings for the world's waterways and elevation postings for the world's terrain. The imagery used is a derivation of the heights from the bathymetry and terrain data.

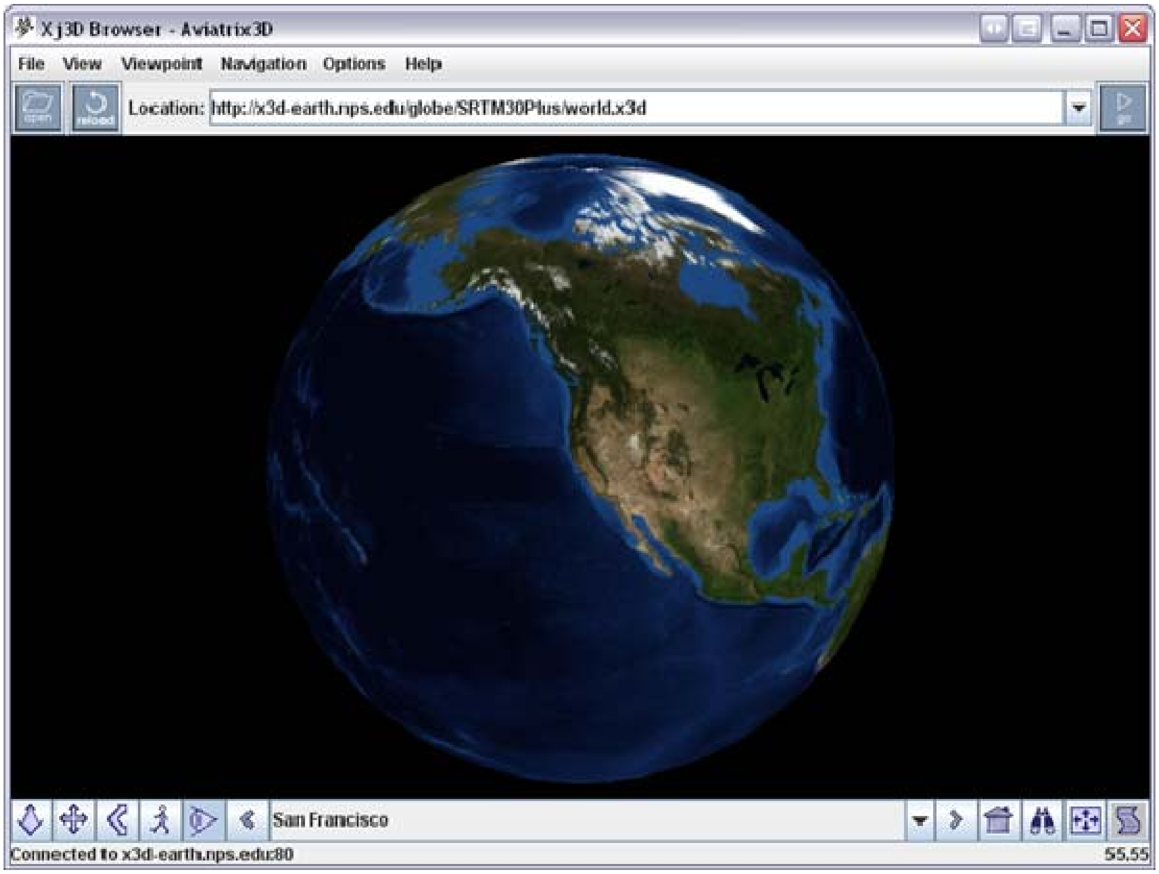


Location: http://x3d-earth.nps.edu/globe/MBARI1MinuteBathy/world.x3d

Connected to x3d-earth.nps.edu:80

Figure 26. MBARI 1-Minute Bathymetry Globe (From X3D-Earth, 2007)

The following example, illustrated in Figure 27, is a low-resolution globe generated using the Terrain Tile Production Chain with Blue Marble Imagery and SRTM 30 Plus terrain data. Both these data sources are freely available on the Internet. The Blue Marble Imagery is 500-meter resolution and the SRTM terrain is 1-kilometer resolution.



Location: http://x3d-earth.nps.edu/globe/SRTM30Plus/world.x3d

San Francisco

Connected to x3d-earth.nps.edu:80

55.55

Figure 27. X3D-Earth Globe, Blue Marble Imagery and SRTM 30 Plus Terrain (From X3D-Earth, 2007)

These low-resolution globes stand as examples of the utility of the X3D-Earth project. They are relatively easy to generate and can be used in a number of applications using a simple X3D node, the *Inline* node. An example of the use of this node is found in Chapter VIII, Section A-1.

3. OpenStreetMap and OpenAerialMap Globes

The final contribution from Dr. Yoo, were the OpenStreetMap and OpenAerialMap globes. These globes are similar to the previous two globes, yet they are generated “on-the-fly” using self-referring PHP scripts to generate the X3D terrain tiles which then reference the needed imagery (Yoo & Brutzman, 2009).

> In order to test initial server-side architecture for X3D-Earth, a self-referring PHP script implementation was created which generates X3D

> terrain-tile sets on-the-fly. The server receives an initial query for location, and then constructs an X3D scene that includes top-level *GeoElevationGrid* height data with appropriate draped *ImageTexture* imagery, along with child URL links to the next-lower quadtree of similarly constructed scenes. The PHP script is then used repeatedly to query, generate and link terrain-tile files dynamically for each subsequent Inline quadtree child.

This method is very similar to the preprocessed method, but does the preprocessing steps automatically. The data for the elevation and imagery are retrieved automatically at runtime and only the needed data is loaded (Yoo & Brutzman, 2009). Below are the links to the resulting globes.

X3D-Earth top-level scene: http://x3d-earth.nps.edu/d0.x3d

Open Street Map X3D scene: http://x3d-earth.nps.edu/osmb0.x3d

Open Aerial Map X3D scene: http://x3d-earth.nps.edu/oamb0.x3d

Videos of Dr. Yoo's globes can be found at http://x3d-earth.nps.edu/video.

Unfortunately, the imagery assets of OpenAerialMap were taken offline in 2009. Current work in progress will restore this important resource. For further information, see http://openaerialmap.org.

H. EXISTING X3D-EARTH GENERATION CODEBASES

1. Rez (Chris Thorne)

Rez is an "open source framework and tools for translating gridded data, mainly geospatial, to different formats including images and multi-resolution models for X3D or VRML web browsing" (Thorne, 2007). The intention of Rez is to provide a tool "for modeling large 3D terrains and whole planets for display over the web" (Thorne, 2007). Rez is written in Java and the source code can be downloaded from http://sourceforge.net/projects/planet-earth. "Rez is designed as a multi-adapter with a central program (Rez.java) which allows the input parser to be plugged in and the output generator to be plugged in" (Thorne, 2007). Rez parses terrain files and outputs them to a single file or a multi-resolution tree of tiles at a resolution and size controlled by the user.

It is designed to be able to handle large datasets, and be flexible enough for users to build their own plugging for use with any format (Rez, 2007).

Rez is an excellent tool for use in this thesis. It is open source and works on a variety of data formats. Its use of command line execution and configuration files is essential in using it for generating globes on the Hamming Supercomputer. Thus, the architecture for building the 3D globes was driven by the capabilities of Rez. The examples provided with the tool are easy to follow and reproducible.

There are two primary input mechanisms into Rez, configuration files and the command-line parameters (or parameters selected from a GUI). These two inputs are what inform Rez which plug-ins are used, the source directories, output directories, and other essential information for properly executing the generation of 3D data. The configuration files are discussed first.

a. Rez Configuration File

This file is an essential piece to running the Rez code. It contains a number of input parameters that generally remain constant, thus they are included in the configuration file vice the command-line parameters. These parameters are listed below with a description of each:

- Source Directory - The directory which contains the input file which is referenced in the TileList parameter. The directory must exist or Rez returns an error.
- Destination Directory - The directory into which Rez writes the generated data. The directory must exist or Rez returns an error.
- Scene - This parameter informs Rez which scene generating plug-in is used.
- GUI - This parameter informs Rez which Graphical User Interface will be used. The Rez source code comes with a SimpleGui class that can be modified for use by the user.
- TileArrays - This parameter informs Rez how many files to expect in the TileList parameter.
- TileXDim and TileYDim - These two parameters inform Rez how many files to expect in the X dimension and Y dimension, respectively.

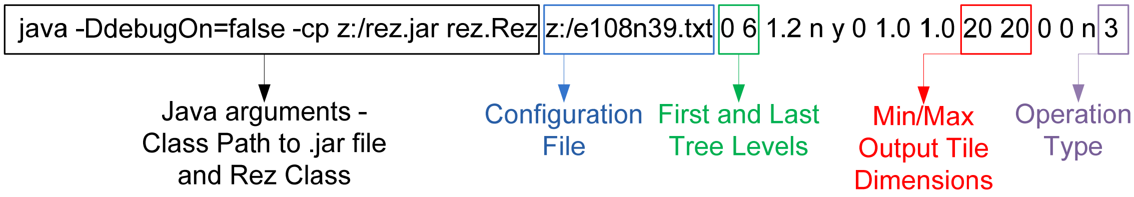
- MaxDecimalPrecision - This parameter informs Rez about the number of decimal points to expect when reading a text file containing the height values for each elevation posting.
- NorthToSouthRows - This parameter informs Rez about the succession of rows in the output elevation grid, and whether they proceed from north to south or south to north.
- sourceDataNorthToSouthRows - This parameter informs Rez about the succession of rows in the input terrain file, and whether they proceed from north to south or south to north.
- Info - This parameter is for additional information intended for Rez. This parameter is generally not used; therefore the string '….' is generally used and is a good placeholder.
- Parser - This parameter informs Rez which parser plug-in will be used to parse the input file.
- Tiler - This parameter informs Rez which tiler plug-in will be used to output the elevation grid.
- TileList - This parameter informs Rez about the file name(s) for the input file(s).

For a detailed description of the format of the configuration file, see Appendix B., which includes an example configuration file.

b. Command-Line Parameters

This section will describe the structure and use of the command-line parameters.

The primary means of executing the Rez codebase is through the use of the command-line. This command-line execution has a specific structure that uses a series of arguments to inform Rez about the other parameters not included in the configuration file. These parameters are listed in the documentation. Below is an example of what values are used in a command-line for running Rez:



```
java -DdebugOn=false -cp z:/rez.jar rez.Rez z:/e108n39.txt 0 6 1.2 n y 0 1.0 1.0 20 20 0 0 n 3
```

Java arguments - Class Path to .jar file and Rez Class
Configuration File
First and Last Tree Levels
Min/Max Output Tile Dimensions
Operation Type

Figure 28. Rez Command-Line Example

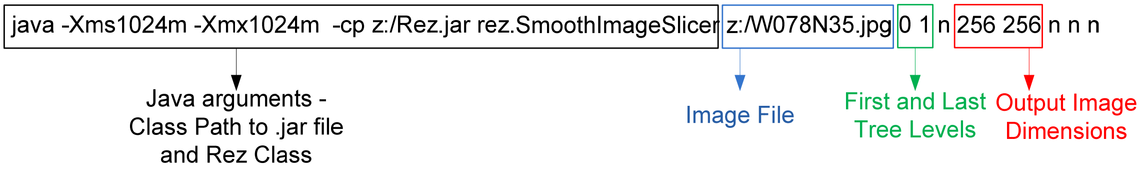
Of all the parameters, only a few were changed from the example files. These are illustrated in Figure 28. The remaining parameters remain the same. The list below contains the parameters that were used and what their function is.

- Configuration File - This parameter contains the absolute location of the configuration file. In the generation of the configuration files and command-line calls, this parameter is matched to each individual configuration file with the terrain file to be processed.
- First/Last Tree Level - These two consecutive parameters inform Rez about the levels of the pyramid. These two parameters remain constant for a given globe generation process, but change from globe to globe.
- Minimum and Maximum Output Tile Dimensions - These parameters inform Rez what the output dimensions will be. This parameter illustrates the number of values that will be output to the elevation grid for both columns and rows. These parameters are generally the same number, to allow a set value for each level in the pyramid. For example, for DTED0 input files, the dimensions were set to 20 and 20. The result is a 20 by 20 *GeoElevationGrid*. Typically the higher the resolution of the input data, the higher the dimensions will.
- Operation Type - This parameter informs Rez about the operation type is chosen for processing the terrain files. The initial architecture had Rez combining thirty adjacent terrain tiles, which would use Operation Type 3, in which “all source tiles are treated as one combined tile which is then split” (Thorne, 2007). The most common and default operation type is Operation Type 1, in which each file is split into an individual pyramid. This type was used in the ultimate architecture refined in this thesis

A detailed description of each parameter can be found in the Rez Documentation at http://www.rez3d.com/Rez.pdf.

c. *ImageSlicer Parameters*

The ImageSlicer class, included in the Rez codebase, is an image manipulation tool that creates levels of images that correspond with the terrain tiles generated by Rez. There are a number of subclasses which represent specific types of image slicers, yet the most commonly used is the SmoothImageSlicer class. Like Rez, the ImageSlicer class can be executed via the command-line, though there is currently no GUI. The command-line parameters are organized in a similar structure as the Rez command-line. Of the parameters only three were changed throughout the process. Below is an example of the command-line for running the ImageSlicer class and subclasses.



```
java -Xms1024m -Xmx1024m  -cp z:/Rez.jar rez.SmoothImageSlicer z:/W078N35.jpg 0 1 n 256 256 n n n
```

Figure 29. ImageSlicer Command-Line Example

The list below contains the parameters there were used and what their functions are.

- Image File - This parameter is the absolute location of the image file.
- First/Last Tree Level - Like the parameters in the Rez command-line, these parameters inform Rez about the levels of the pyramid. These parameters match the levels of the corresponding Rez-generated globe.
- Output Image Dimensions - These parameters inform Rez about the ultimate dimensions of the output images generated by the image slicer. These parameters will remain constant for a given globe, but could change with higher or lower resolution imagery.

A detailed description of each parameter can be found in the Rez Documentation at http://www.rez3d.com/Rez.pdf.

2. Tile Set Manager Application Programming Interface (tsmApi)

Tile Set Manager Application Programming Interface (tsmApi) is “an Open Source library of high level C functions for reading, writing, and processing the terrain data used by the TerraVision terrain visualization application” (tsmApi Library). The tsmApi library is bridge between the terrain data and the API. The terrain data and imagery are stored in multi-resolution tile sets that are organized in databases that organize the tile sets by name and location (tsmApi Library).

The tsmApi has VRML97 and some GeoVRML support in the generation of 3D terrain models. The tsmApi supports multiple types of imagery and terrain file formats, including USGS DOQs, TIFF, GeoTIFF, JPEG, GIF for imagery and USGS 7.5-minute DEMs, DTED Level 0, raw binary grids for terrain (tsmApi Library). The following Figure illustrates the input and output features of tsmApi.

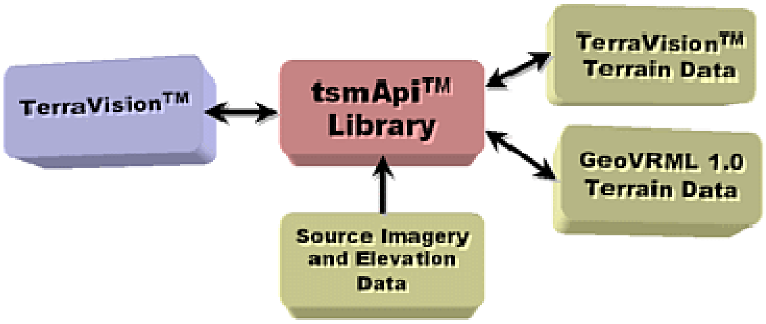


Figure 30. tsmApi Input/Output Diagram (From tsmApi Library, 2002)

Figure 31 illustrates the use of the multi-resolution tile sets. Each successive level of detail has higher resolution per image. The lowest level of detail has the lowest resolution and only one image for the location, and is located at the top of the pyramid tree. The highest level of detail has the highest resolution and 64 images to cover the location, and is located at the bottom of the pyramid tree.

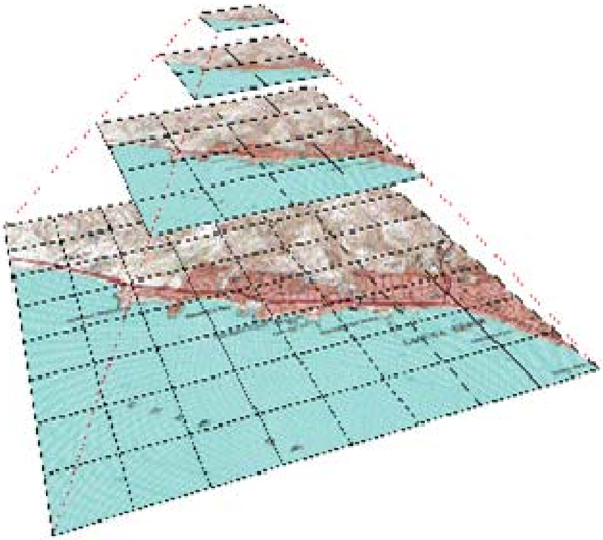


Figure 31. tsmApi Multi-resolution Tile Set Diagram (From tsmApi Library, 2002)

More detailed information and access to the codebase can be found at the Web site, http://www.ai.sri.com/tsmApi/welcome.shtml.

3. Others

There are a number of other code bases for generating geospatial 3D models. Though these code bases are not used or discussed in this thesis they deserve mentioning.

a. JeoSpace

Developed by Yumetech, JeoSpace is a Java-based open-source program. You can find documentation and download the source code at www.jeospace.org.

b. DNC to X3D

Doug Maxwell, researcher at Naval Undersea Warfare Center, has produced a code base for converting unclassified DNC charts to X3D-Earth scenes.

I. KML

1. Purpose and Capability

Keyhole Markup Language, KML, is an XML file format that displays a variety of features in a geospatial model. KML is an open standard format that is maintained by the OGC. KML can display image overlays, pushpins, routes, and other geospatially references data on a 3D rendered globe. KML is displayed in a number of browsers and display formats from Google Earth, Google Maps, to NASA World Wind among others. The full interactive reference for all elements of the KML Schema can be found at http://code.google.com/apis/kml/documentation/kmlreference.html.

2. X3D-Edit Support

Further enhanced as part of this thesis work, X3D-Edit gives a KML developer a fully capable KML GUI. It offers a complete array of Editing Panel support for each KML element. The process for accessing these Editing Panels is similar to accessing the X3D Editing Panels by dragging and dropping the element from the Palette or by right clicking the element once it is in the file. Another feature is that X3D-Edit validates the KML files to the KML schema. The full KML Schema can be found at http://schemas.opengis.net/kml/2.2.0/ogckml22.xsd.

3. KML Usability with X3D-Earth Players

a. XSLT Stylesheet Transformation KmlToX3d.xslt

The KmlToX3d Stylesheet transforms the KML file to an X3D file. This is another capability of X3D-Edit. The KML file can be transformed by right clicking within the file and selecting XSLT Transformation. Currently there is limited capability to transform the KML files. The KmlToX3dViewpoint.xslt Stylesheet transforms KML locations to X3D GeoViewpoints. This is quite useful when multiple locations need to be viewed in succession. The stylesheet can be found at www.web3d.org/x3d/content/examples/Basic/GeoSpatial/KmlToX3dViewpoint.xslt.

b. *X3D Prototypes for KML*

The “X” in X3D stands for eXtensible, and the primary node that makes X3D so extensible is the Prototype node. This node allows an author to write his/her own node. The author is able to customize every in the node, the name, fields, attributes, etc. This new node can then be used an unlimited amount of times. The primary purpose of the Prototype is to create and test new nodes prior to being permanently incorporated into the X3D language.

Currently, there are two X3D prototypes for KML, PlaceMark and Point. The PlaceMark allows the user to place a PlaceMark in geospatial space with only two coordinates, latitude and longitude. A number of other attributes are needed to properly position the PlaceMark, such as GeoSystem and others common to Geospatial components. The prototype displays a balloon which is clamped to the ground no matter what the terrain elevation might be. The Point prototype allows the user to display a 3D point in any location by using three coordinates, latitude, longitude and altitude. The prototypes can be found at http://www.web3d.org/x3d/content/examples/Basic/GeoSpatial/KmlPrototypes.html.

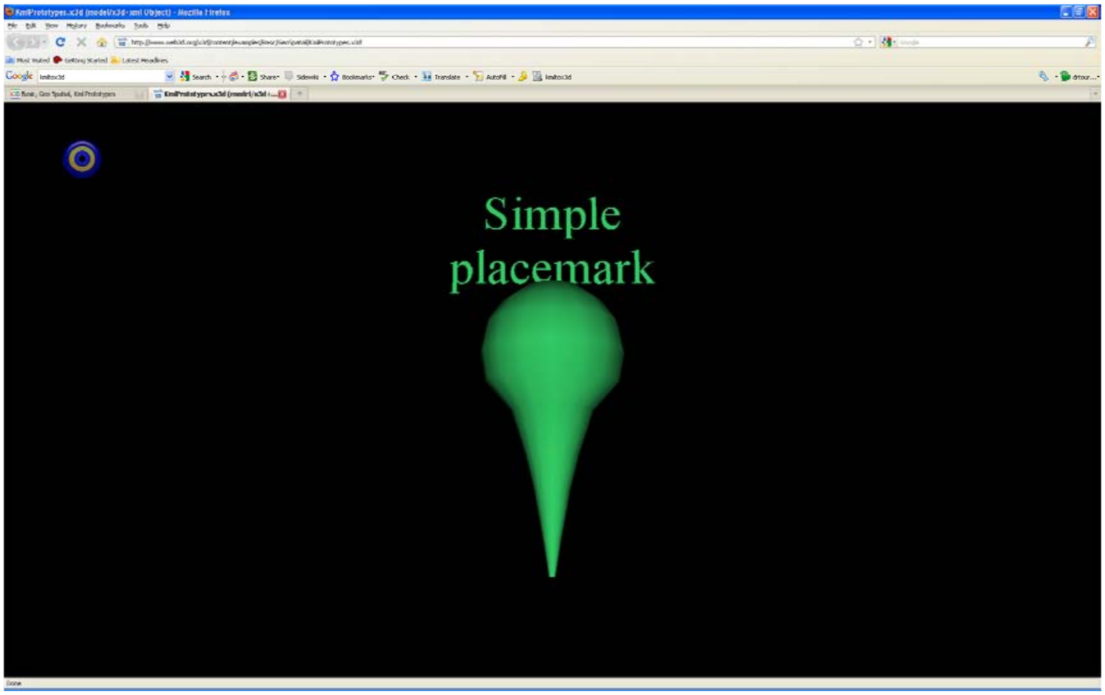

KmlPrototypes.x3d (model/x3d+xml Object) - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Most Visited Getting Started Latest Headlines
Search Share Sidewiki Bookmarks Check Translate AutoFill
Simple
placemark

Figure 32. Screenshot of KML Placemark in X3D

c. Embedding X3D in KML

This section discusses embedding X3D in KML files. KML currently has the capability of importing 3D models into a browser. This is accomplished by using a <Link> reference to a model on the Web inside of the <Model> node within a KML file. A URL is used in the <href> element of the <Link>. The model is positioned within the overall model by the <Location> element. The <Model> node can also orient and scale the model, by use of those elements within the node. Currently, Collada is most commonly used for this capability, but the KML model can also use X3D models.

d. Potential Native Support and Mixed Namespaces for KML in X3D

There is potential within X3D to expand for native use of KML. There are proposed changes for X3D Version 3.3 to include Namespaces for KML within the specification. This would allow KML nodes to be used natively within X3D models. This is a useful expansion for geospatial models, since KML is widely used throughout the world for representing geospatial data. This allows users to easily switch from another format or browser to X3D and vice versa.

J. SUMMARY

The above chapter covers the prior work that has advanced the X3D-Earth project. It describes the foundation work, established by Greg Leaver while a thesis student at NPS. It also describes the advances made by Brian Hittner while at NPS. The chapter describes the progress achieved by Dr. Chris Thorne's dissertation and his open-source codebase, Rez. It also describes Dr. Yoo's contributions while a postdoctorial research associate at NPS. It describes the other X3D-Earth scene generating codebases. Finally, it describes KML and how it can be used with X3D.

IV. AVAILABLE IMAGERY, CARTOGRAPHY, TERRAIN, AND BATHYMETRY

A. INTRODUCTION

National Geospatial-Intelligence Agency (NGA) provides all of the data used for this thesis. The process for obtaining this data is detailed in Chapter VII, Section A. The following are the types of data obtained for this thesis.

B. DATA USED

1. Imagery

a. NaturalVue

"NaturalVue-Rev2 product consisting of complete global coverage of natural color-enhanced Landsat digital data (acquired circa year 2000), which has been precision ortho-rectified and seamlessly mosaiced" (Dykstra, n.d.). The resolution for the NaturalVue imagery is 0.5 arc seconds or approximately 15m resolution at the equator. The imagery can be obtained from NGA in two file formats, JPEG-2000 and TIF.

For this thesis, both formats were requested and received. The JPEG-2000 files are much smaller than the GeoTIFF files, since they are compressed. The GeoTIFF files are not compressed. Thus, each file is broken into quad sets with each file being over one gigabyte. The total size of the JPEG-2000 files is approximately 197 gigabytes, while the GeoTIFF files are over 4 terabytes.

There are some issues with dealing with these two formats. For JPEG-2000, there is limited support in Java. For the GeoTIFF files, the primary issue is dealing with the large sizes of the files and the irregular dimensions of the individual files within the quad set. These issues are detailed further in Chapter VI, Section D-3, for JPEG-2000, and Section D-4 for GeoTIFF.

Figure 33 shows the coverage of the NaturalVue imagery. Each green square represents a single file in JPEG-2000 format or a quad set in GeoTIFF format.

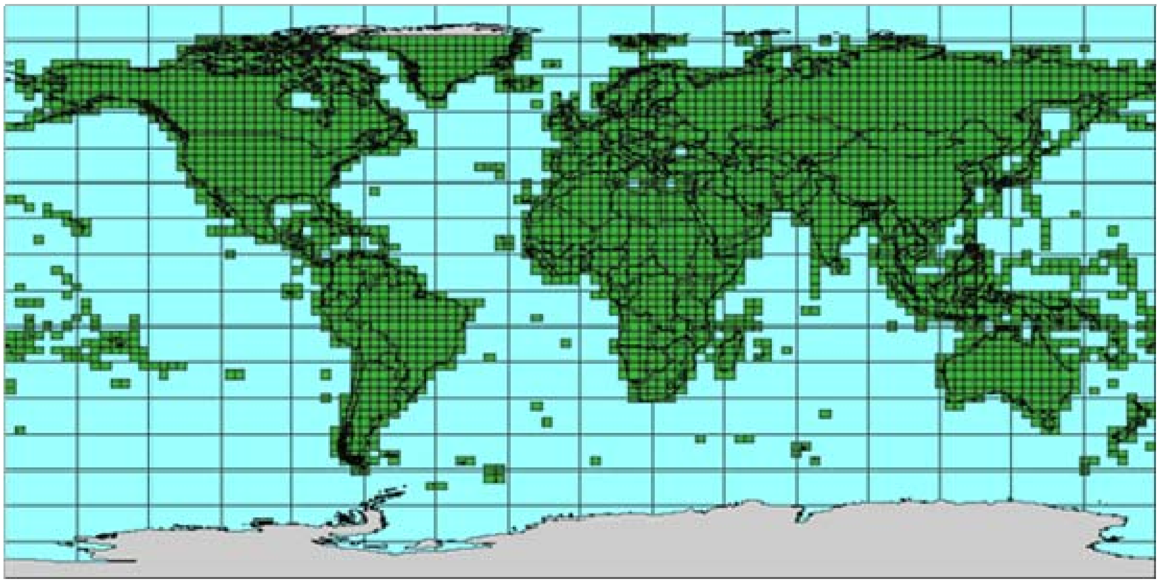


Figure 33. NaturalVue Coverage (From Dykstra, n.d.)

b. Blue Marble

> In 2002, NASA produced the Blue Marble, the most detailed true-color image of the Earth's surface ever produced. Using data from NASA's Terra satellite, scientists and data visualizers stitched together four months of observations of the land surface, coastal oceans, sea ice, and clouds into a seamless, photo-like mosaic of every square kilometer (.386 square mile) of our planet. In October 2005, NASA released a new version of the spectacular image collection that provides a full year's worth of monthly observations with twice the level of detail as the original. The new collection is called the Blue Marble: Next Generation. (History of Blue Marble, n.d.)

Blue Marble Next Generation has been used as a basis for many low resolution globes produced for X3D-Earth. This thesis uses this imagery as a basis for the low resolution globes. The resolution for the imagery is 500 meters per pixel. Figure 34 is a sample of 500 meters resolution imagery.

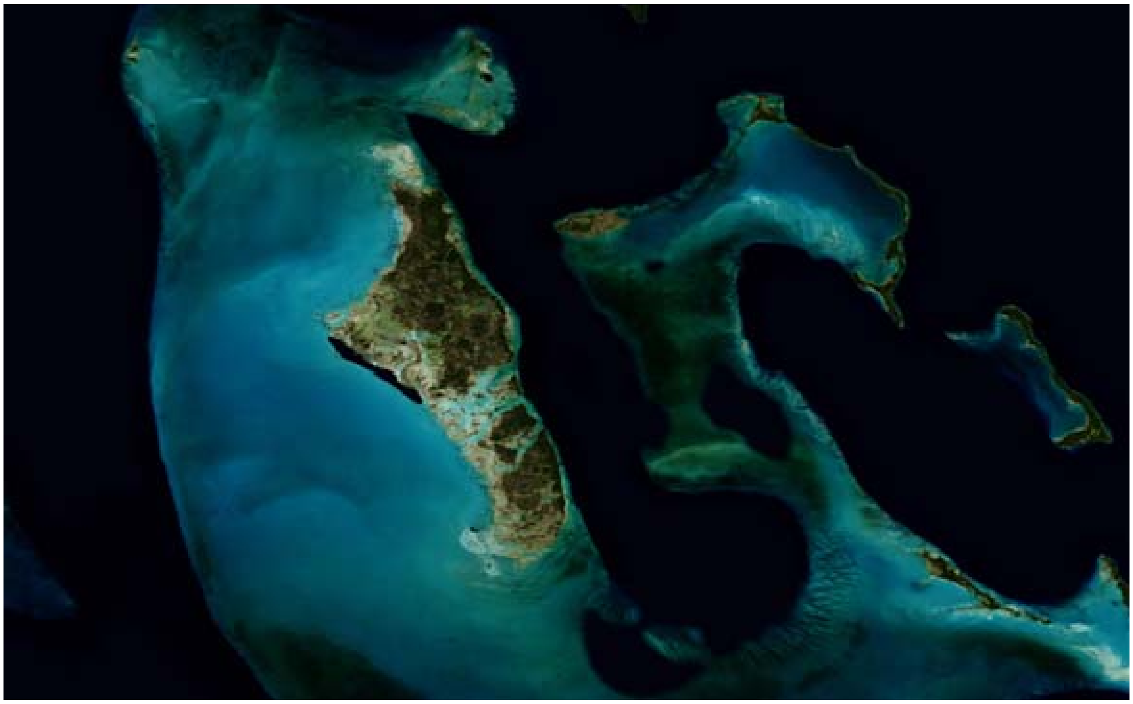


Figure 34. 500-Meter Resolution Image of the Bahamas From Blue Marble. Image from Institute for Atmospheric and Climate Science.

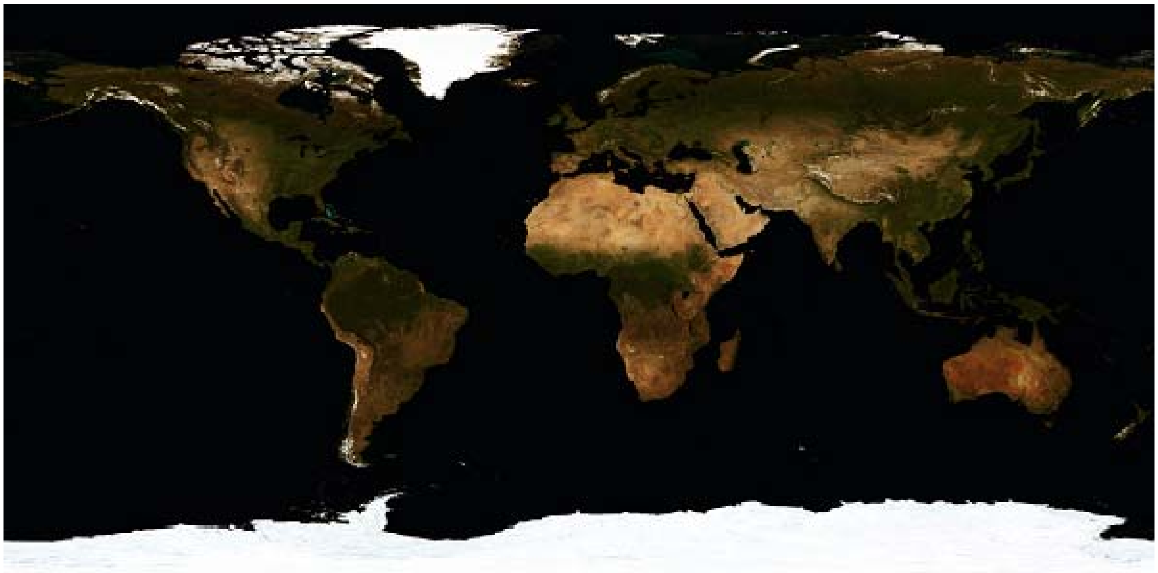


Figure 35. Blue Marble Imagery (From Stockli et al., 2005)

Figure 35 is a full illustration of the Blue Marble coverage. A full download of all months and composite Imagery can be found at http://earthobservatory.nasa.gov/Features/BlueMarble/BlueMarble_monthlies.php.

2. Terrain

a. Digital Terrain Elevation Data (DTED)

Digital Terrain Elevation Data is terrain data available from many publically accessible locations on the Internet. The resolution of the data is 30 arc seconds, which is approximately one kilometer. It is derived from the DTED Level 1 data, and its primary purpose is to be widely distributed and for public use. DTED Level 0 is meant for “scientific, technical, and other communities for applications that require terrain elevation, slope, and/or surface roughness information” (Digital Terrain Elevation Data, 2008). DTED Level 1 is a medium resolution data set that is used “for all military activities and systems that require landform, slope, elevation, and/or gross terrain roughness in a digital format” (Digital Terrain Elevation Data, 2008). DTED Level 1 is at 3 arc second or 100 meter resolution. DTED Level 2 is the high resolution terrain data that is at 1 arc second or thirty meters resolution.

DTED is the primary source of terrain for this thesis. The initial globe created for this thesis consists of DTED Level 0 data. The initial goal was to create a globe with a mix of DTED Levels 0, 1, and 2 to produce maximum precision. Yet the task of creating the initial globe proved too difficult.

The terrain data can be obtained from NGA in compact disk form or loaded on an external hard drive. More information can be found at https://www1.nga.mil/ProductsServices/TopographicalTerrestrial/DigitalTerrainElevationData/Pages/default.aspx.

C. DATA OBTAINED BUT NOT USED

1. Imagery

a. Controlled Image Base (CIB) Database

Controlled Image Base is a seamless imagery dataset produced by NGA. It comes in three resolutions, 1, 5 and 10 meters. The 10 meter resolution imagery is the original CIB format and is now obsolete (NGA, 2008). Following the 10-meter resolution data, NGA began producing the 5 meter data, and now relies on the 1 meter data. The 5-meter data is still produced and available, though the 1-meter data is in higher demand by Mission Planners (NGA, 2008).

The imagery comprises "grayscale (panchromatic) images [that] are projected onto the Arc System, ortho-rectified, and assembled into a mosaic" (NGA, 2008). Color imagery is now available. This imagery is not used in this thesis, but the imagery was obtained and can be used for future work. More information is found at https://www1.nga.mil/ProductsServices/TopographicalTerrestrial/ControlledImageBase/Pages/default.aspx.

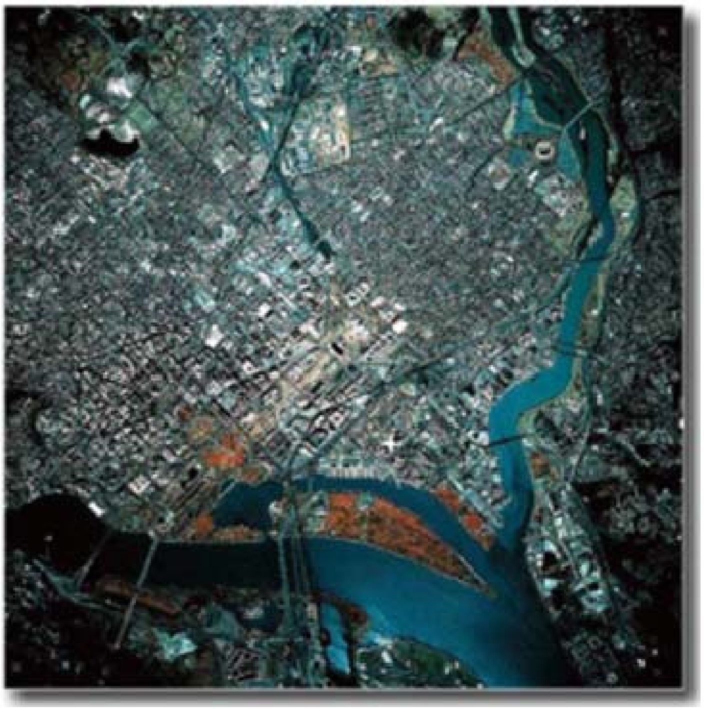


Figure 36. CIB Representation of Washington, D.C. (From NGA, 2008)

2. Cartography

a. Compressed ARC Digitized Radar Graphics (CADRG)

Compressed ARC Digitized Radar Graphics (CADRG) is a digitized version of maps and charts used in military applications. CADRG is derived from the ADRG and other maps and charts which have been converted to Raster Product Format (RPF) for easier use in computer programs. Further information about CADRG can be found in the specifications at http://jitc.fhu.disa.mil/nitf/tag_reg/docs/89038_cadrg.pdf.

This dataset is not used in this thesis but was obtained and can be used in future work.

b. Digital Nautical Charts (DNC)

"The Digital Nautical Chart (DNC) is a vector based digital product that is designed to provide the mariner with an up-to-date seamless database of the world" (NGA, n.d.). "The Digital Nautical Chart database is based on and developed from the feature content of NGA and National Ocean Service hard copy charts. The features depicted are thematically organized into 12 layers or coverages including: Cultural Landmarks, Earth Cover, Environment, Hydrography, Inland Waterways, Land Cover, Limits, Aids to Navigation, Obstructions, Port Facilities, Relief, and Data Quality" (NGA, n.d.).

More information can be found at http://www.nga.mil/NGAPortal/DNC.portal.

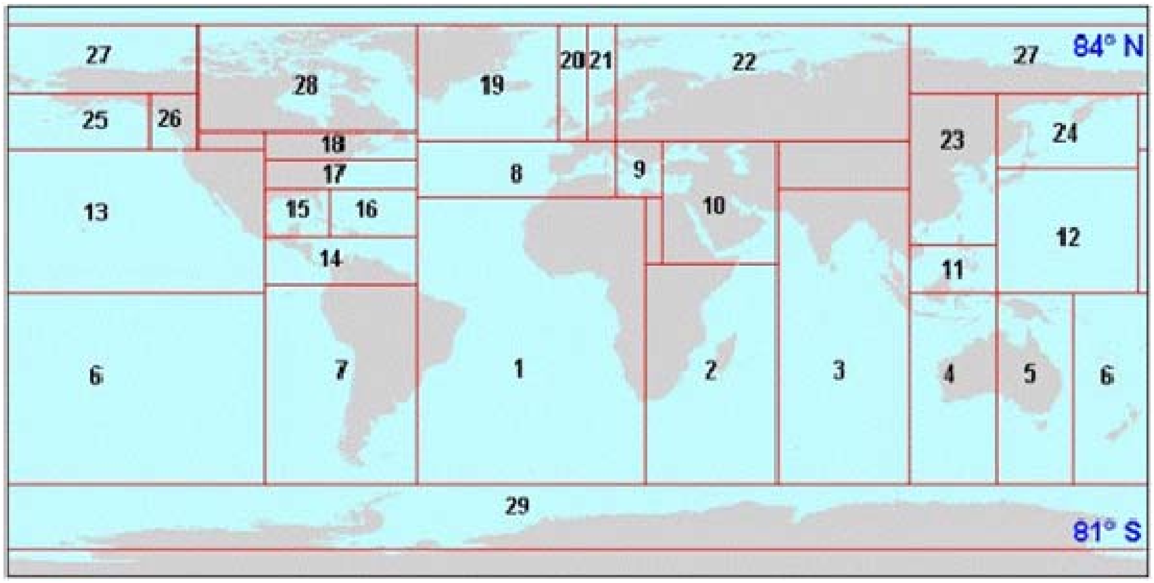


Figure 37. DNC CD Coverage. Image From National Geospatial-Intelligence Agency, http://www.nga.mil/NGAPortal/DNC.portal.

The DNC charts were obtained from NGA but are not used in this thesis.

3. Bathymetry

a. Digital Nautical Charts (DNC)

DNCs have a bathymetry component within each chart file. These soundings can be extracted to produce an XYZ file for use in Rez. This method is not thoroughly explored in this thesis, but initial attempts are made using the Global Mapper software to export the sounding component as an XYZ file. Once this file is exported, Rez has the capability to use the file as a basis for building a *GeoElevationGrid* for the soundings. Doug Maxwell uses this bathymetry in his X3D-Earth generator. Another option for extracting this component from the charts is to write a program to do so and store it in an XYZ format.

b. MBARI 1-Minute Bathymetry

Though this bathymetry was not directly used in this thesis, it deserves mentioning. MBARI has produced an extensive dataset of the Earth’s oceans and land at a resolution of 1-minute or approximately 1.8 kilometers. This dataset was used in developing some of the earlier low resolution X3D-Earth globes. This dataset was developed and compiled by Mike McCann, and later used in an X3D-Earth model developed by Dr. Byounghyun Yoo. This data was not used in this thesis.

D. SUMMARY

The above chapter describes the various types of data obtained in this thesis process. It describes the imagery used, Blue Marble and NaturalVue. It describes the cartography obtained, CADRG and DNC. The chapter describes the terrain obtained and used, DTED and SRTM. Finally, it describes the bathymetry obtained, DNC and MBARI 1- Minute Bathymetry.

V. X3D GEOSPATIAL COMPONENT AND X3D-EARTH DISTRIBUTED SCENE GRAPH

A. INTRODUCTION

This chapter describes the X3D Geospatial Component and all its nodes, as well as X3D-Edit support for the Geospatial Component. Finally, it describes the X3D-Earth distributed scene graph generated by Rez.

B. X3D GEOSPATIAL COMPONENT

This section covers the Geospatial component nodes for X3D. The *GeoElevationGrid* and *GeoLOD* nodes are important for this thesis and the others are listed with summaries. The specification for all X3D Geospatial component nodes can be found at http://www.web3d.org/x3d/specifications/ISO-IEC-19775-1.2-X3D-AbstractSpecification/Part01/components/geodata.html.

1. GeoElevationGrid

The *GeoElevationGrid* node is similar to the regular X3D *ElevationGrid* node, yet it is geographically referenced. This node creates "a height field where all coordinates are specified in terms of [coordinates - depending on geoSystem], and elevation" Geospatial Component, 2008). This height field is geographically referenced based on the specified *geoSystem* field value. The *geoSystem* field determines what type of coordinate system is to be used.

Values for *GeoElevationGrid* node are generated by the Rez codebase. Rez parses the input terrain file and generates the user requested levels of detail at the specified dimensions. In each file generated by Rez there is a *GeoElevationGrid* node that contains the elevation values for that specified level of detail. Since each level of the quad tree has four times as many files as the prior level, for example, in the level 0 file there is one file that contains a *GeoElevationGrid* node that contains elevation values at the specified dimensions, while at level 1 there are four files each with a single GeoElevationGrid representing a higher level of detail of elevation values.

2. GeoLOD

The *GeoLOD* node is the most important node within the X3D Geospatial Component for this thesis. Its proper utilization is what makes the quad tree pyramid format work.

The *GeoLOD* node is much like the *LOD* node, yet it is geographically referenced. The node allows the user to specify two levels of detail to be rendered. The range attribute is used for determining which level of detail will be rendered. The geometry within the *GeoLOD* or referenced in *rootUrl* or *rootNode* is rendered when the viewer is outside of the given range. When inside the given range the browser renders the geometry represented in the *child1Url*, *child2Url*, etc. Below in Figure 38, is a pictorial representation of the *GeoLOD* node.

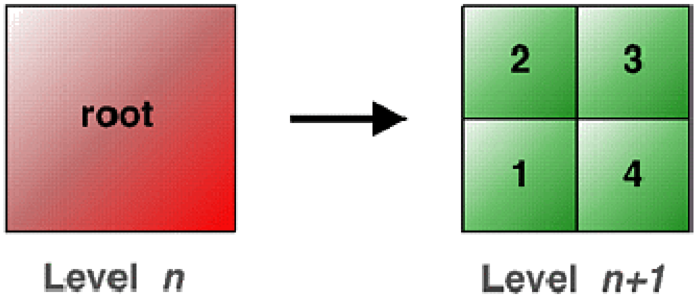
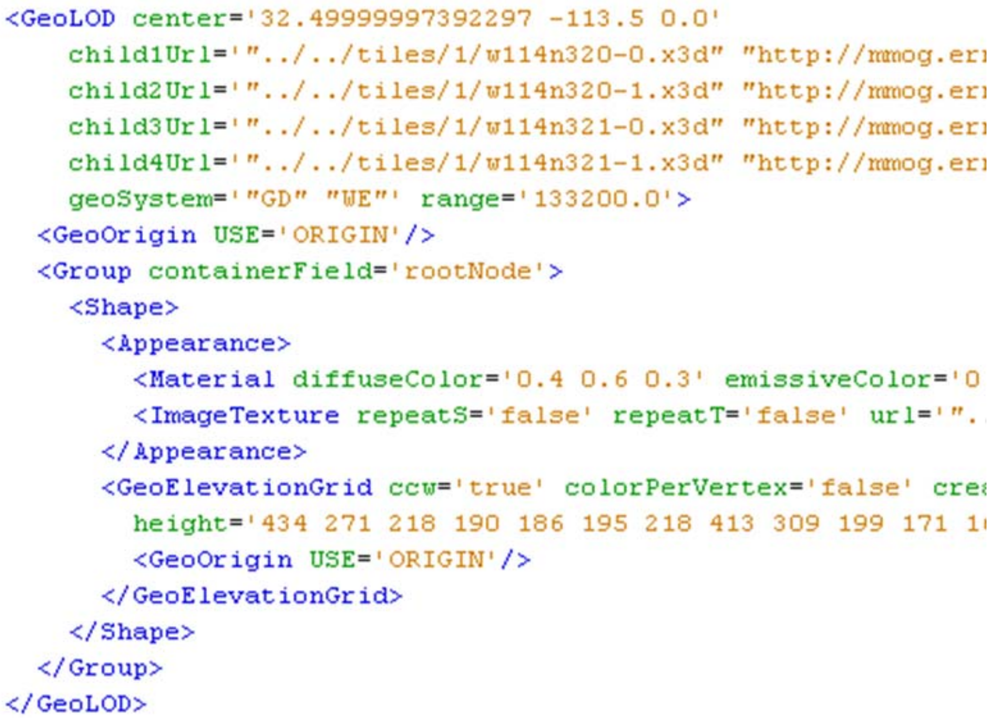


Figure 38. Loading of GeoLOD Levels (From Geospatial Component, 2008)

For this thesis, Rez generates the X3D files that contain these nodes. An integral part is the use of the *GeoLOD* node. In the Rez generated files the *GeoElevationGrid* node, explained in Section A-1, is the *rootNode.* The elevation included in that *GeoElevationGrid* is represented in the user specified. The child1Url through child4Url represents URL references to the next level of detail. For example, if the user generates a terrain pyramid from level 0 to level 2 at ten by ten dimensions, the top most file at level 0 will contain one file with a node that *GeoElevationGrid* contains one hundred elevation values and a *GeoLOD* node that points to the four files of level 1. These four files will contain a *GeoElevationGrid* node of one hundred elevation values and a

GeoLOD that points to the level 2 files. The final level will contain the sixteen level 2 files with each a *GeoElevationGrid* node. The Figure 39 shows an example of the *GeoLOD* node.



```
<GeoLOD center='32.49999997392297 -113.5 0.0'
    child1Url='"../../tiles/1/w114n320-0.x3d" "http://mmog.er
    child2Url='"../../tiles/1/w114n320-1.x3d" "http://mmog.er
    child3Url='"../../tiles/1/w114n321-0.x3d" "http://mmog.er
    child4Url='"../../tiles/1/w114n321-1.x3d" "http://mmog.er
    geoSystem='"GD" "WE"' range='133200.0'>
  <GeoOrigin USE='ORIGIN'/>
  <Group containerField='rootNode'>
    <Shape>
      <Appearance>
        <Material diffuseColor='0.4 0.6 0.3' emissiveColor='0
        <ImageTexture repeatS='false' repeatT='false' url='".
      </Appearance>
      <GeoElevationGrid ccw='true' colorPerVertex='false' cre
        height='434 271 218 190 186 195 218 413 309 199 171 1
        <GeoOrigin USE='ORIGIN'/>
      </GeoElevationGrid>
    </Shape>
  </Group>
</GeoLOD>
```

Figure 39. Example of the GeoLOD Node

3. GeoViewpoint

The *GeoViewpoint* node allows the user to place a viewpoint within the scene that is in geospatial coordinates. The *GeoViewpoint* has the same fields and behaviors as the standard Viewpoint in X3D, with three exceptions. These are the *geoOrigin*, *geoSystem*, and the position. The *geoOrigin* states the local coordinate frame for increased precision. The *geoSystem* states which specific datum is used, i.e., WGS84. The position is the actual coordinates the viewpoint is located in the correct format for the given datum.

4. GeoOrigin

The *GeoOrigin* node states what the absolute geospatial location is, and provides a local reference frame for translating and rotating the geometry. The *geoCoords* field states the actual position of the origin. The *geoSystem* states which datum is to be used.

5. GeoLocation

The *GeoLocation* node allows a user to position an ordinary X3D model in a geospatial world. The *geoOrigin* and *geoSystem* fields are the same as in other Geospatial nodes. The *geoCoords* field is similar, yet it can be changed dynamically by an exterior node. The children field holds the model that is positioned by this node. All children are oriented to the up direction for that local area (the normal to the tangent plane on the ellipsoid) (Web3D, 2008).

6. GeoPositionInterpolator

The *GeoPositionInterpolator* is an interpolator for geospatial nodes. It has a *geoOrigin* and *geoSystem* like the other Geospatial nodes. The other fields are similar to the *PositionInterpolator* node, yet the *keyValue* field contains geospatial coordinates. The *geovalue_changed* field can output the value via a ROUTE node to a *GeoLocation* and *GeoViewpoint* node for animation.

7. GeoTransform

The *GeoTransform* node is similar to the Transform node, yet it translates and rotates in a geospatial reference frame. The *geoCenter* is similar to the *center* field yet it is geospatially referenced. The *geoOrigin* and *geoSystem* fields are identical to the other geospatial nodes. All children are oriented to the up direction for that local area.

8. GeoProximitySensor

The *GeoProximitySensor* is similar to the *ProximitySensor* node, yet it is based on a geospatial reference frame. The node generates events when the user enters, exits or moves within a geospatially referenced bounding box. These events are expressed in a *geoCoords_changed* field that is returned when the viewer moves into, out of, or within the box. This field can be output into other nodes.

9. GeoTouchSensor

The *GeoTouchSensor* is the same as a *TouchSensor,* yet it is geospatially referenced. An added capability to the original *TouchSensor* is that the *GeoTouchSensor* can return the coordinates of the pointing device in the geospatial reference frame.

C. X3D-EDIT AUTHORING SUPPORT

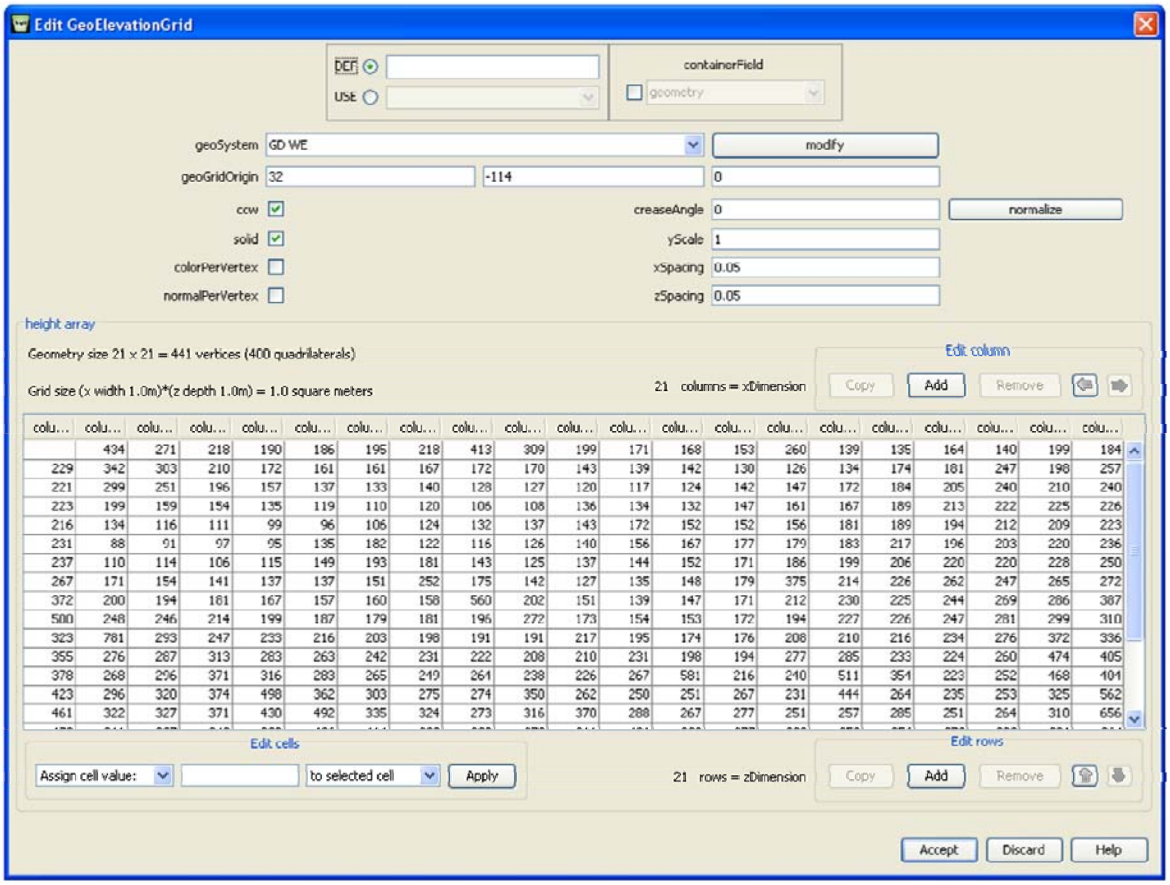
This section presents the X3D-Edit Authoring support for the X3D Geospatial Component.

1. X3D Geospatial Component Editing Panels

X3D-Edit contains user friendly Editing Panels for every node available in the X3D language. Among these nodes are the X3D Geospatial nodes. These Editing Panels increase the usability of the language by acting as a guide to users. These panels are accessed by dragging and dropping a new node from the Palette, and by right clicking on the node and selecting Edit. Once the Editing Panel is open it allows the user to see and edit the given attributes and fields of the nodes. The Editing Panels also have self-checking ability to guarantee the user has entered the appropriate data in the appropriate format. This occurs when exiting the Editing Panel. In the following section, a brief description of each Editing Panel will be described for each node.

a. GeoElevationGrid

The *GeoElevationGrid* Editing Panel allows the user to input the *geoSystem* field by typing it in or using the drop down menu. The *geoOrigin* is set by entering the x, y, and z values in the respective blocks. The other major component of the *GeoElevationGrid* is the Grid. There are buttons to add rows and columns, and once these are added the user can enter in the data for each gridded square. Once the attributes and fields are set, the Accept button adds the node or edits the attributes and fields. Below is a snapshot of the generic *GeoElevationGrid* Editing Panel.



DEF ●

USE ○

containerField ☐ geometry

geoSystem GD WE

geoGridOrigin 32 -114 0

ccw ☑

solid ☑

colorPerVertex ☐

normalPerVertex ☐

creaseAngle 0

yScale 1

xSpacing 0.05

zSpacing 0.05

height array

Geometry size 21 x 21 = 441 vertices (400 quadrilaterals)

Grid size (x width 1.0m)*(z depth 1.0m) = 1.0 square meters

21 columns = xDimension

colu...	colu...	colu...	colu...	colu...	colu...	colu...	colu...	colu...	colu...	colu...	colu...	colu...	colu...	colu...	colu...	colu...	colu...	colu...	colu...	colu...
	434	271	218	190	186	195	218	413	309	199	171	168	153	260	139	135	164	140	199	184
229	342	303	210	172	161	161	167	172	170	143	139	142	130	126	134	174	181	247	198	257
221	299	251	196	157	137	133	140	128	127	120	117	124	142	147	172	184	205	240	210	240
223	199	159	154	135	119	110	120	106	108	136	134	132	147	161	167	189	213	222	225	226
216	134	116	111	99	96	106	124	132	137	143	172	152	152	156	181	189	194	212	209	223
231	88	91	97	95	135	182	122	116	126	140	156	167	177	179	183	217	196	203	220	236
237	110	114	106	115	149	193	181	143	125	137	144	152	171	186	199	206	220	220	228	250
267	171	154	141	137	137	151	252	175	142	127	135	148	179	375	214	226	262	247	265	272
372	200	194	181	167	157	160	158	560	202	151	139	147	171	212	230	225	244	269	286	387
500	248	246	214	199	187	179	181	196	272	173	154	153	172	194	227	226	247	281	299	310
323	781	293	247	233	216	203	198	191	191	217	195	174	176	208	210	216	234	276	372	336
355	276	287	313	283	263	242	231	222	208	210	231	198	194	277	285	233	224	260	474	405
378	268	296	371	316	283	265	249	261	238	226	267	581	216	240	511	351	223	252	468	401
423	296	320	374	498	362	303	275	274	350	262	250	251	267	231	444	264	235	253	325	562
461	322	327	371	430	492	335	324	273	316	370	288	267	277	251	257	285	251	264	310	656

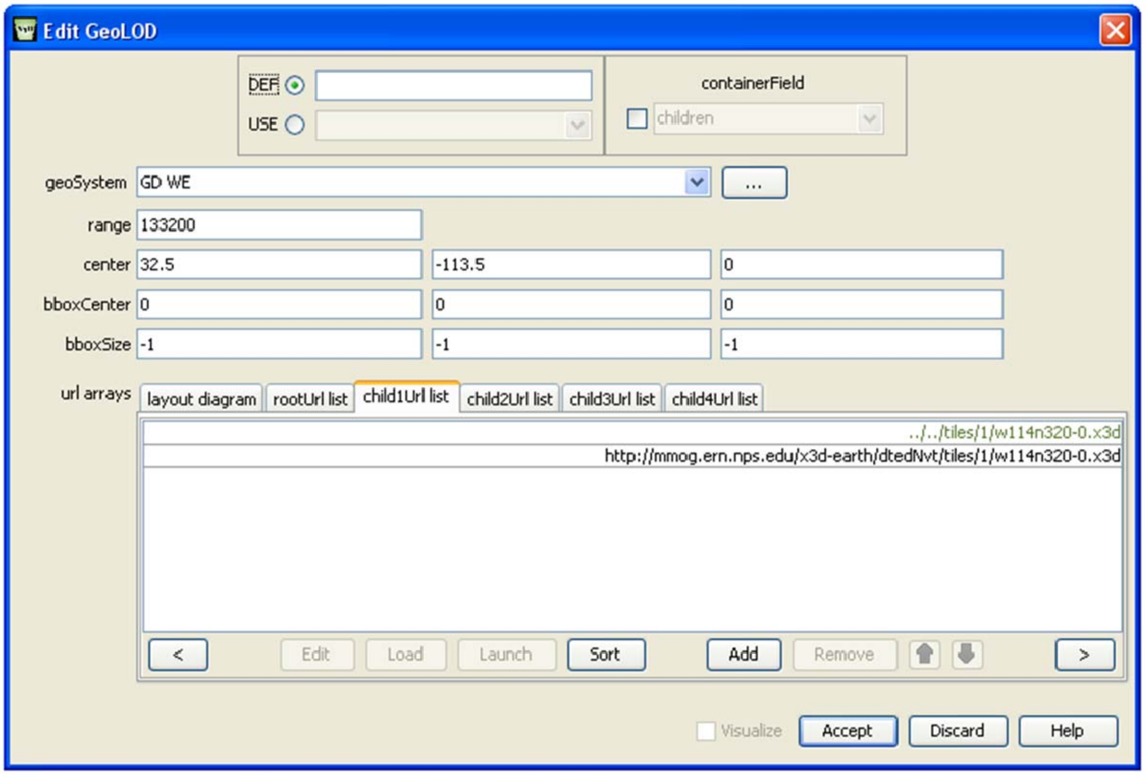
Assign cell value: to selected cell

21 rows = zDimension

Figure 40. *GeoElevationGrid* Editing Panel in X3D Edit

b. *GeoLOD*

The *GeoLOD* Editing Panel allows the user to input the *geoSystem* field by typing it in or using the drop down menu. The *range* is set by entering the z value in box provided; this sets the range at which the geometry rendered changes from the root node or scene referenced in the *rootUrl* to the children geometry/scenes referenced in the child1Url, child2Url, etc. The *center* field is set by entering the x, y, and z values in the respective boxes. The other major component of the *GeoLOD* is the various URLs used. The *rootUrl* is the scene that is to be rendered when the position of the viewer is at a distance from the *center* that is greater than the range. The other URLs that need to be set are the *child1Url*, *child2Url*, etc. The URLs reference the scenes that will be rendered when the viewer is closer than the range field to the *geoOrigin* of the node. Each of these fields has their own tab. Once these tabs are selected an Add or Remove button is present to add or drop URLs. If the Add button is selected a new window is opened that allows the user to enter the URL or browse to find the file. Once the attributes and fields are set, the Accept button adds the node or edits the attributes and fields. Figure 41 is a snapshot of the *GeoLOD* Editing Panel.



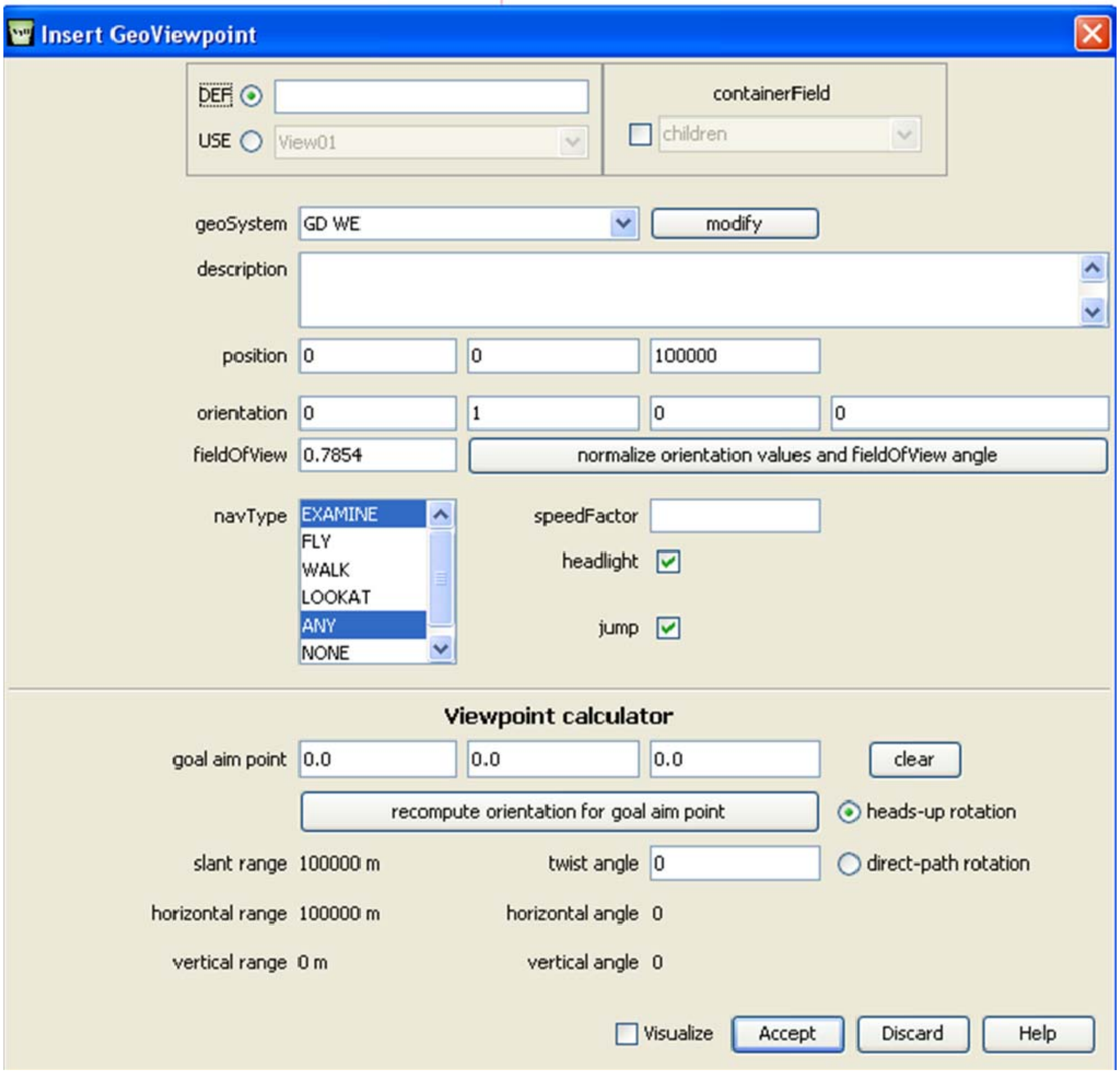
DEF ●
USE ○
containerField □ children
geoSystem GD WE
range 133200
center 32.5 -113.5 0
bboxCenter 0 0 0
bboxSize -1 -1 -1

../../tiles/1/w114n320-0.x3d
http://mmog.ern.nps.edu/x3d-earth/dtedNvt/tiles/1/w114n320-0.x3d

Figure 41. *GeoLOD* Editing Panel in X3D Edit

c. GeoViewpoint

The *GeoViewpoint* Editing Panel is identical to the *Viewpoint* Editing Panel with the exceptions of the *geoSystem* and *navType* attributes. The *GeoViewpoint* Editing Panel allows the user to input the *geoSystem* field by typing it in or using the drop down menu. A *description* can be entered to let other users know what the viewpoint is intended to be. The *position* field is set by entering the x, y, and z values in the respective boxes. The *orientation* field is set by entering the x, y, and z values of the orientation axis, ranging from -1 to 1. The fourth value in the set is the rotation angle in radians about that axis. The navType panel allows the user to determine which navigation type the scene will have from that viewpoint. Finally there is a viewpoint calculator that allows the user to input the x, y, and z of a target object to look at and then cut and paste if desired. Once the attributes and fields are set, the Accept button adds the node or edits the attributes and fields. Below is a snapshot of the generic *GeoViewpoint* Editing Panel.


Insert GeoViewpoint
modify
normalize orientation values and fieldOfView angle
clear
recompute orientation for goal aim point
Accept
Discard
Help

DEF ●
USE ○ View01

containerField □ children

geoSystem GD WE

description

position 0 0 100000

orientation 0 1 0 0

fieldOfView 0.7854

navType EXAMINE, FLY, WALK, LOOKAT, ANY, NONE

speedFactor

headlight ☑

jump ☑

Viewpoint calculator

goal aim point 0.0 0.0 0.0

● heads-up rotation

○ direct-path rotation

slant range 100000 m

twist angle 0

horizontal range 100000 m

horizontal angle 0

vertical range 0 m

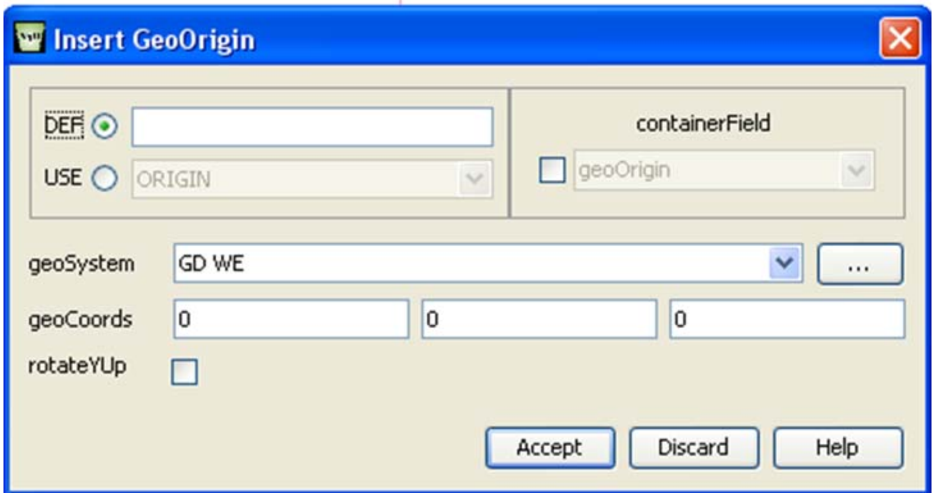
vertical angle 0

□ Visualize

Figure 42. *GeoViewpoint* Editing Panel in X3D Edit

d. *GeoOrigin*

The *GeoOrigin* Editing Panel allows the user to input the *geoSystem* field by typing it in or using the drop down menu. The *geoCoords* field is set by entering the x, y, and z values in the respective boxes. There is also a check box that allows the user to select whether or not to rotate with the y up. Once the attributes and fields are set, the Accept button adds the node or edits the attributes and fields. Figure 43 is a snapshot of the generic *GeoOrigin* Editing Panel.

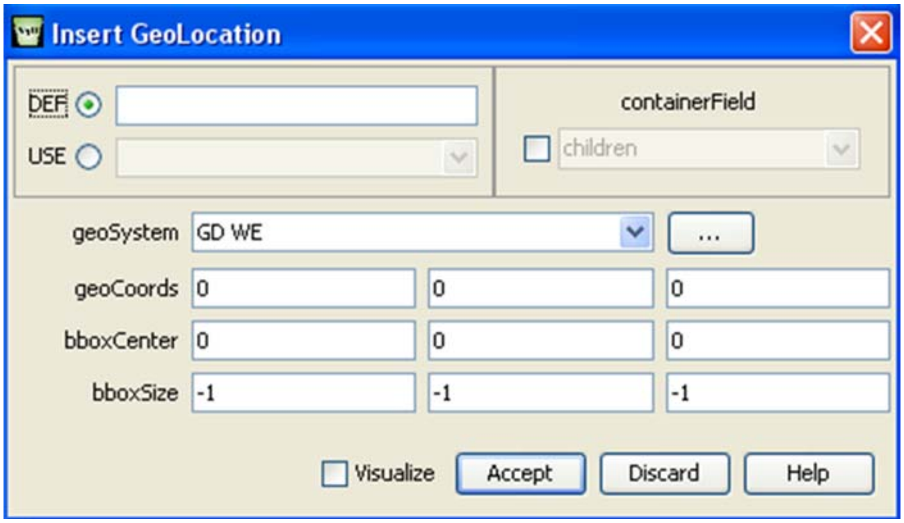


DEF ●
USE ○ ORIGIN
containerField □ geoOrigin
geoSystem GD WE
geoCoords 0 0 0
rotateYUp □

Figure 43. *GeoOrigin* Editing Panel in X3D Edit

e. *GeoLocation*

The *GeoLocation* Editing Panel allows the user to input the *geoSystem* field by typing it in or using the drop down menu. The *geoCoords* field is set by entering x, y, and z values in the respective boxes. The *bboxCenter* represents the Bounding Box center and there are three boxes for entering the x, y, and z coordinates. The *bboxSize* represents the size of the Bounding Box and there are three entry slots for entering the size values in x, y, and z planes in meters. There is also a check box that allows the user to select whether or not to visualize the *GeoLocation*. This will add geometry to the scene to render the Bounding Box at the user defined location. Once the attributes and fields are set, the Accept button adds the node or edits the attributes and fields. Figure 44 is a snapshot of the generic *GeoLocation* Editing Panel.

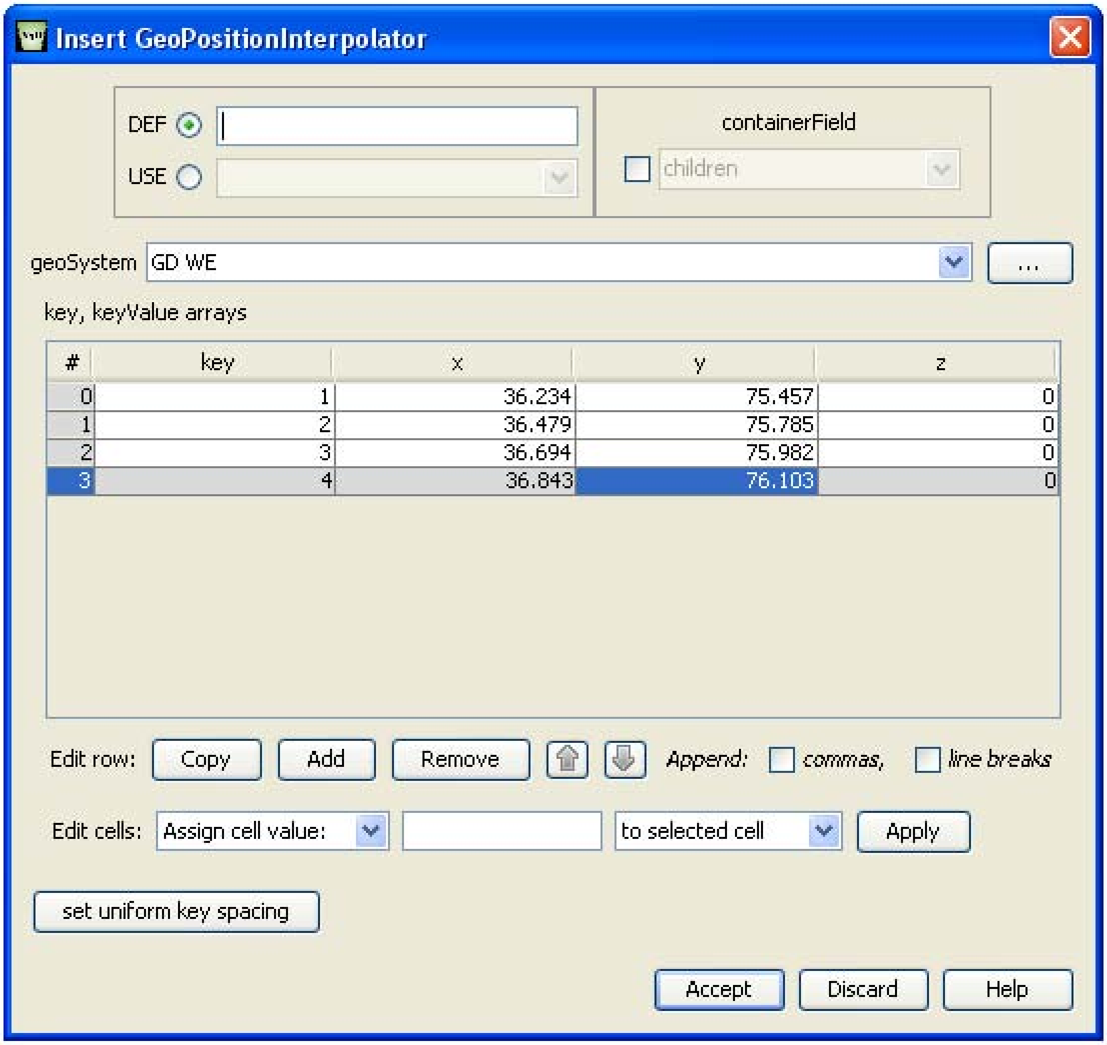


● DEF
○ USE
□ children
geoSystem GD WE
geoCoords 0 0 0
bboxCenter 0 0 0
bboxSize -1 -1 -1
□ Visualize

Figure 44. *GeoLocation* Editing Panel in X3D Edit

f. GeoPositionInterpolator

The *GeoPositionInterpolator* Editing Panel is identical to the *PositionInterpolator* with the *geoSystem* attribute added. GeoPositionInterpolator allows the author to define a list of latitude/longitude geographic positions. The *GeoPositionInterpolator* Editing Panel allows the user to input the *geoSystem* field by typing it in or using the drop down menu. Once the attributes and fields are set, the Accept button adds the node or edits the attributes and fields. Figure 45 is a snapshot of the generic *GeoPositionInterpolator* Editing Panel.



DEF ● USE ○

containerField □ children

geoSystem GD WE

key, keyValue arrays

#	key	x	y	z
0	1	36.234	75.457	0
1	2	36.479	75.785	0
2	3	36.694	75.982	0
3	4	36.843	76.103	0

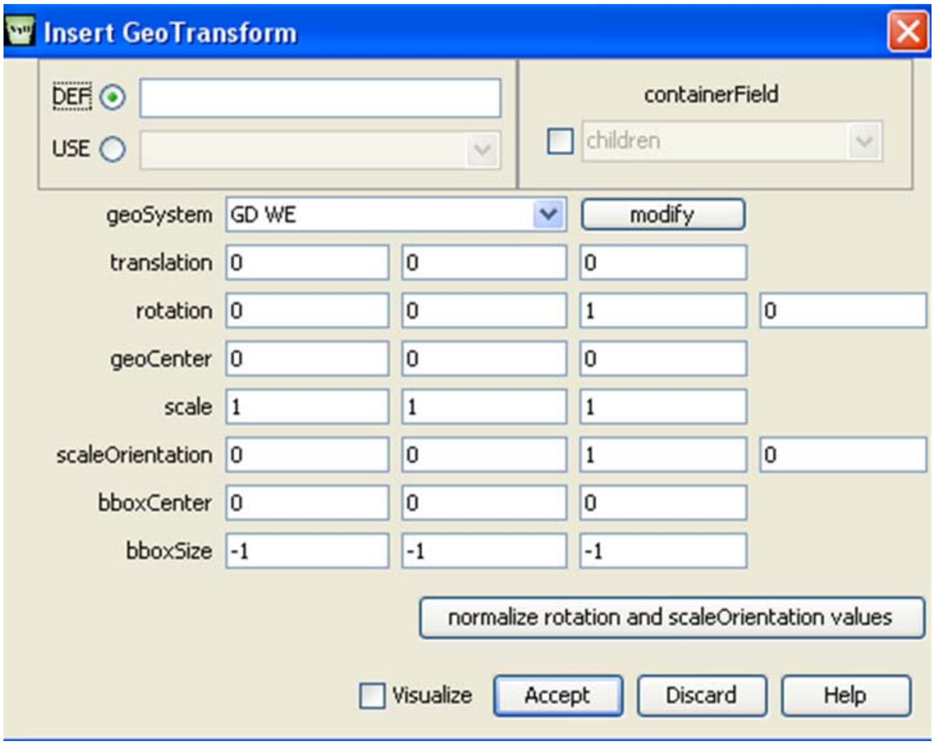
Edit row: Append: □ commas, □ line breaks

Edit cells: Assign cell value: to selected cell

Figure 45. *GeoPositionInterpolator* Editing Panel in X3D Edit

g. GeoTransform

The *GeoTransform* Editing Panel is similar to the *Transform* Editing Panel with the *geoSystem* added and a *geoCenter* instead of a center. It allows the user to input the *geoSystem* field by typing it in or using the drop down menu. The remaining attributes and fields are set like in the *Transform* node. Once the attributes and fields are set, the Accept button adds the node or edits the attributes and fields. Figure 46 is a snapshot of the generic *GeoTransform* Editing Panel.



Insert GeoTransform

DEF ●

USE ○

containerField □ children

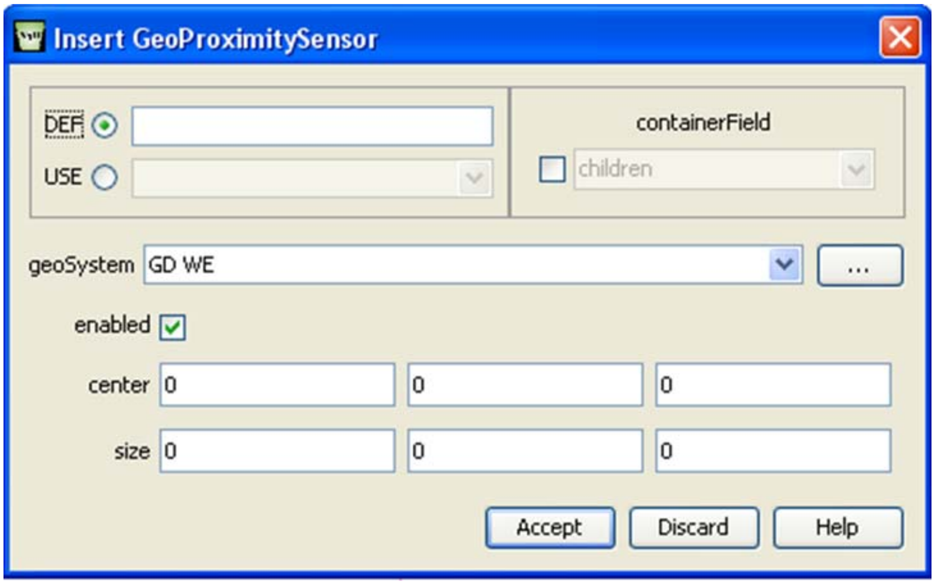
geoSystem	GD WE			
translation	0	0	0	
rotation	0	0	1	0
geoCenter	0	0	0	
scale	1	1	1	
scaleOrientation	0	0	1	0
bboxCenter	0	0	0	
bboxSize	-1	-1	-1	

□ Visualize

Figure 46. *GeoTransform* Editing Panel in X3D Edit

h. GeoProximitySensor

The *GeoProximitySensor* Editing Panel is similar to the *ProximitySensor* Editing Panel with the *geoSystem* added. It allows the user to input the *geoSystem* field by typing it in or using the drop down menu. The remaining attributes and fields are set like in the *ProximitySensor* node. Once the attributes and fields are set, the Accept button adds the node or edits the attributes and fields. Figure 47 is a snapshot of the generic *GeoProximitySensor* Editing Panel.



Insert GeoProximitySensor

DEF ●

USE ○

containerField □ children

geoSystem GD WE

enabled ☑

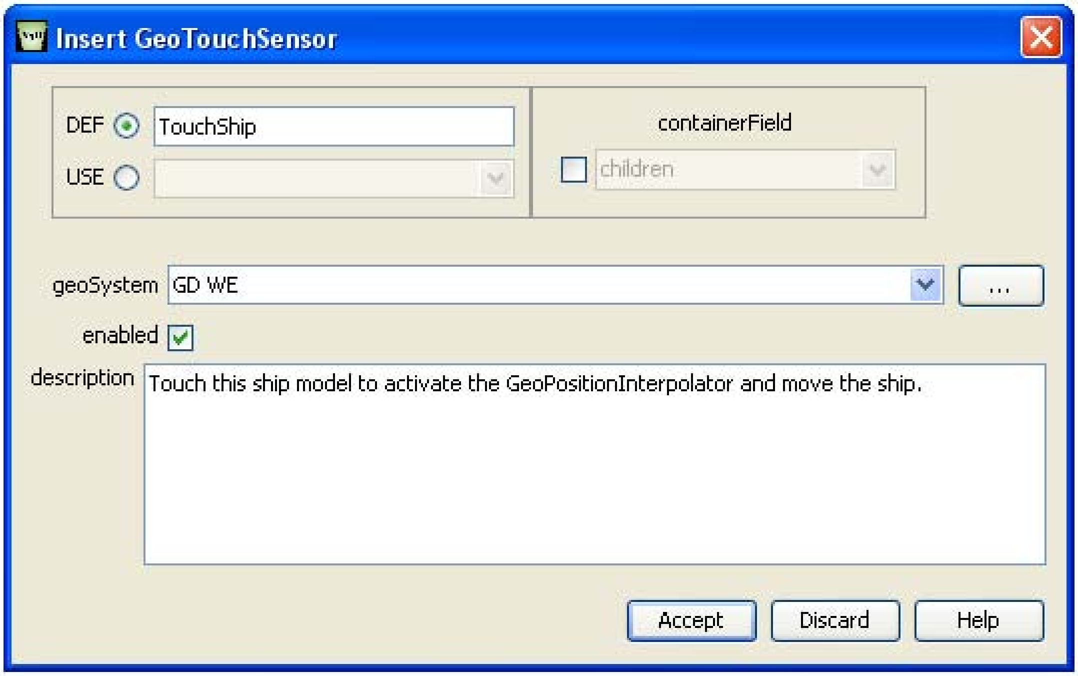
center 0 0 0

size 0 0 0

Figure 47. *GeoProximitySensor* Editing Panel in X3D Edit

i. GeoTouchSensor

The *GeoTouchSensor* Editing Panel is similar to the *TouchSensor* Editing Panel with the *geoSystem* added. It allows the user to input the *geoSystem* field by typing it in or using the drop down menu. The remaining attributes and fields are set like in the *TouchSensor* node. Once the attributes and fields are set, the Accept button adds the node or edits the attributes and fields. Figure 48 is a snapshot of the generic *GeoTouchSensor* Editing Panel.



Insert GeoTouchSensor

DEF ● TouchShip

USE ○

containerField

□ children

geoSystem GD WE

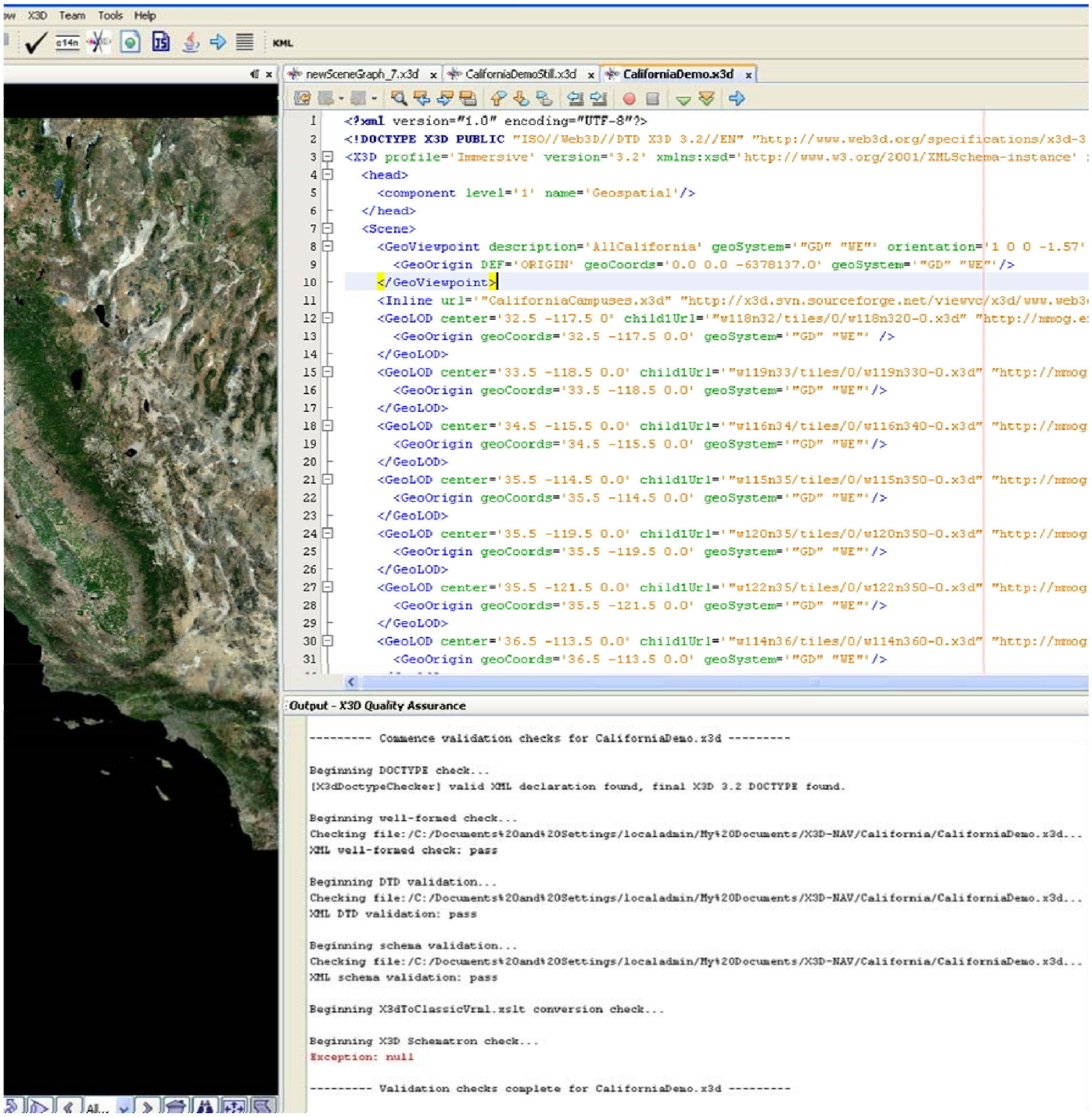
enabled ☑

description Touch this ship model to activate the GeoPositionInterpolator and move the ship.

Figure 48. *GeoTouchSensor* Editing Panel in X3D Edit

2. Quality Assurance (QA)

The Quality Assurance capabilities of X3D-Edit are wide ranging and cover every possible check that can be performed on an XML file. With one click of the button the user can ensure that the file is a Well-formed XML file, validate the file to the XML DTD, validate the file to the XML Schema, convert the file to VRML and validate it, and finally validate the file to the X3D Schematron. Figure 49 is a snapshot of the Quality Assurance check on a Demo X3D file for this thesis.



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.2//EN" "http://www.web3d.org/specifications/x3d-3
<X3D profile='Immersive' version='3.2' xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance' :
  <head>
    <component level='1' name='Geospatial'/>
  </head>
  <Scene>
    <GeoViewpoint description='AllCalifornia' geoSystem='"GD" "WE"' orientation='1 0 0 -1.57'
      <GeoOrigin DEF='ORIGIN' geoCoords='0.0 0.0 -6378137.0' geoSystem='"GD" "WE"'/>
    </GeoViewpoint>
    <Inline url='"CaliforniaCampuses.x3d" "http://x3d.svn.sourceforge.net/viewvc/x3d/www.web3
    <GeoLOD center='32.5 -117.5 0' child1Url='"w118n32/tiles/0/w118n320-0.x3d" "http://mmog.e:
      <GeoOrigin geoCoords='32.5 -117.5 0.0' geoSystem='"GD" "WE"' />
    </GeoLOD>
    <GeoLOD center='33.5 -118.5 0.0' child1Url='"w119n33/tiles/0/w119n330-0.x3d" "http://mmog
      <GeoOrigin geoCoords='33.5 -118.5 0.0' geoSystem='"GD" "WE"'/>
    </GeoLOD>
    <GeoLOD center='34.5 -115.5 0.0' child1Url='"w116n34/tiles/0/w116n340-0.x3d" "http://mmog
      <GeoOrigin geoCoords='34.5 -115.5 0.0' geoSystem='"GD" "WE"'/>
    </GeoLOD>
    <GeoLOD center='35.5 -114.5 0.0' child1Url='"w115n35/tiles/0/w115n350-0.x3d" "http://mmog
      <GeoOrigin geoCoords='35.5 -114.5 0.0' geoSystem='"GD" "WE"'/>
    </GeoLOD>
    <GeoLOD center='35.5 -119.5 0.0' child1Url='"w120n35/tiles/0/w120n350-0.x3d" "http://mmog
      <GeoOrigin geoCoords='35.5 -119.5 0.0' geoSystem='"GD" "WE"'/>
    </GeoLOD>
    <GeoLOD center='35.5 -121.5 0.0' child1Url='"w122n35/tiles/0/w122n350-0.x3d" "http://mmog
      <GeoOrigin geoCoords='35.5 -121.5 0.0' geoSystem='"GD" "WE"'/>
    </GeoLOD>
    <GeoLOD center='36.5 -113.5 0.0' child1Url='"w114n36/tiles/0/w114n360-0.x3d" "http://mmog
      <GeoOrigin geoCoords='36.5 -113.5 0.0' geoSystem='"GD" "WE"'/>
```

```
--------- Commence validation checks for CaliforniaDemo.x3d ---------

Beginning DOCTYPE check...
[X3dDoctypeChecker] valid XML declaration found, final X3D 3.2 DOCTYPE found.

Beginning well-formed check...
Checking file:/C:/Documents%20and%20Settings/localadmin/My%20Documents/X3D-NAV/California/CaliforniaDemo.x3d...
XML well-formed check: pass

Beginning DTD validation...
Checking file:/C:/Documents%20and%20Settings/localadmin/My%20Documents/X3D-NAV/California/CaliforniaDemo.x3d...
XML DTD validation: pass

Beginning schema validation...
Checking file:/C:/Documents%20and%20Settings/localadmin/My%20Documents/X3D-NAV/California/CaliforniaDemo.x3d...
XML schema validation: pass

Beginning X3dToClassicVrml.xslt conversion check...

Beginning X3D Schematron check...
Exception: null

--------- Validation checks complete for CaliforniaDemo.x3d ---------
```

Figure 49. Quality Assurance Check

a. DTD and Schema Validation

X3D-Edit validates the file against the X3D DTD and X3D Schema. These two files contain information on how X3D nodes are to be formed within the XML file format. This process compares the given X3D file to the DTD and Schema independently. If the file can be validated to both the DTD and Schema, the nodes within the file conform to the prescribed formats. If it does not, X3D-Edit reports which nodes are not conforming to the prescribed format and gives a detailed description of how it is incorrect.

The X3D DTD can be found at http://www.web3d.org/specifications/x3d-3.2.dtd.

The X3D Schema can be found at http://www.web3d.org/specifications/x3d-3.2.xsd.

b. X3D Schematron Rule Checking

There is an additional check to the two previously mentioned checks. It is the X3D Schematron Rule Checking. “Schematron is a language for making assertions about patterns and internal consistency found in XML documents” (Brutzman, 2010). The X3D file is compared to these pattern and consistency rules. If the file has nodes that do not conform to these rules error or warning messages are reported. “These quality-assurance checks go beyond the capabilities of DTD or XML Schema grammar-based checking” (Brutzman, 2010). Examples include:

- DEF must precede USE
- DEF and USE cannot be within the same node
- Nodes that do not contain description fields, i.e., Timesensor, should not have one included

3. SMAL Metadata for Terrain Files

Developed by LCDR Travis Rauch, while a student at NPS, SAVAGE Metadata Analysis Language (SMAL) is a language that standardizes the modeling and simulation metadata for any model that is created as a part of the SAVAGE Archive. The goal of SMAL is to standardize the use of metadata for these models so that they can be accessed and referenced more efficiently across a number of visualization platforms. The reasoning behind the need for this is that if a model conforms to SMAL for its metadata, the details of the model can be determined without the need for human interaction after the metadata is coded. Figure 50 is an illustration of the role of SMAL.

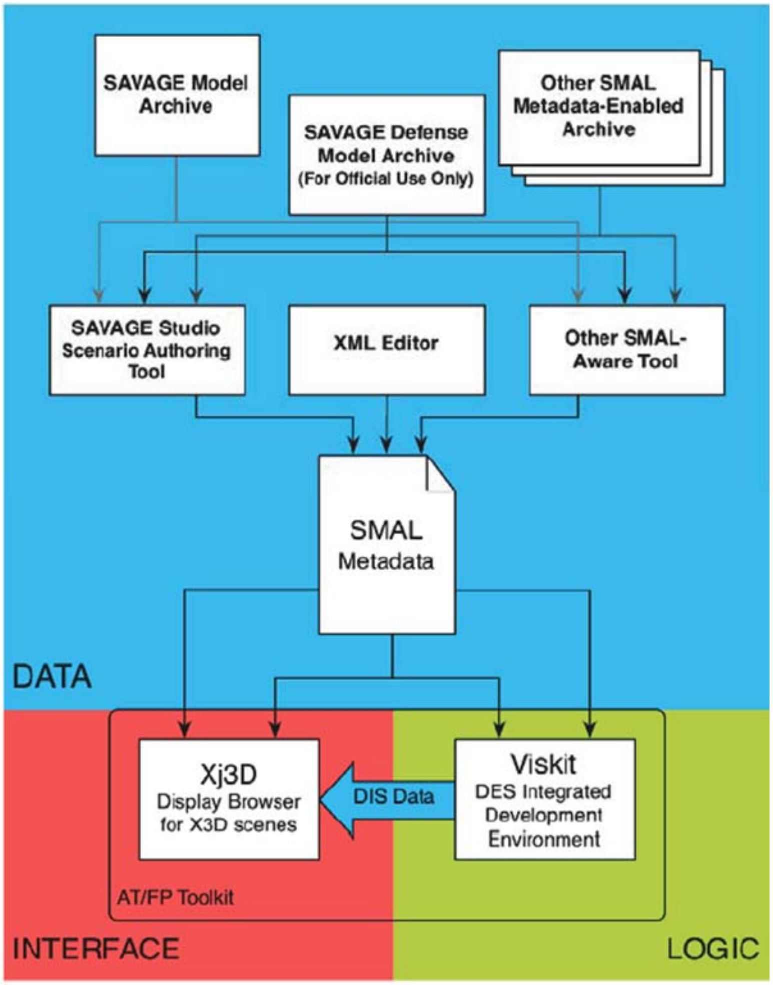

SAVAGE Model Archive
SAVAGE Defense Model Archive (For Official Use Only)
Other SMAL Metadata-Enabled Archive
SAVAGE Studio Scenario Authoring Tool
XML Editor
Other SMAL-Aware Tool
SMAL Metadata
DATA
Xj3D Display Browser for X3D scenes
DIS Data
Viskit DES Integrated Development Environment
AT/FP Toolkit
INTERFACE
LOGIC

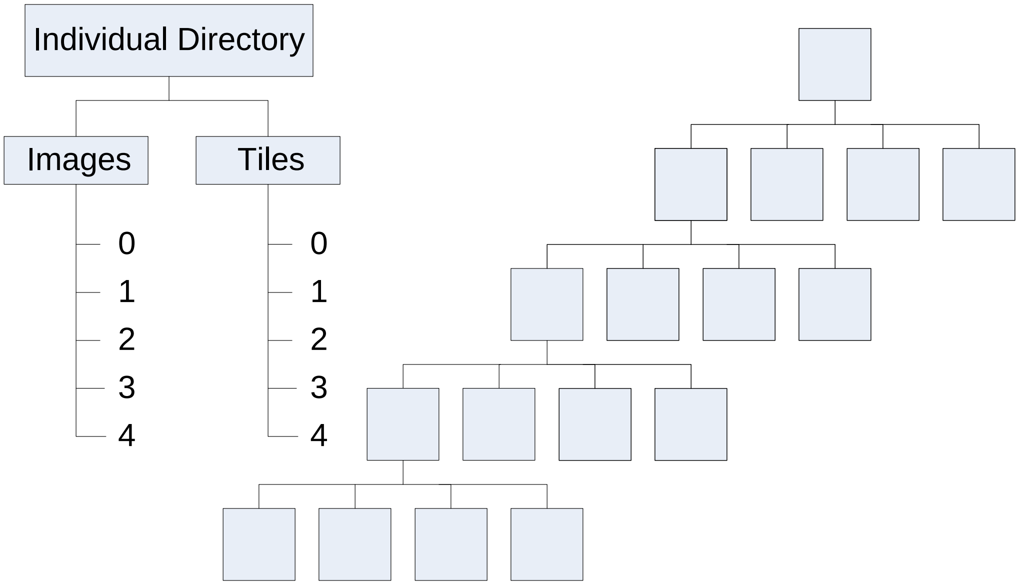
Figure 50. Role of SMAL (From Blais & Norbraten, n.d)

D. X3D-EARTH DISTRIBUTED SCENE GRAPH

1. Rez Generated X3D-Earth Scenes

This section will describe the process in which Rez generates the X3D tiles and the method for creating the quad tree pyramids. As described in the examples of the previous sections, Rez generates the GeoElevationGrid nodes and the GeoLOD nodes that make up the X3D-Earth Distributed scene graph.

The X3D Scene Graph plug-in for Rez provides this structure. Rez creates the chain of URL connections that enable the browser to navigate from the lowest level of detail to the highest level of detail. The "distributed" part of the scene graph comes from the use of multiple files to represent the levels of detail. In each parent file, there is a URL reference to the four higher levels of detail children, i.e., level 0 is the parent of the level 1 file. This reference is created in the GeoLOD node. The Rez X3D Scene Graph plug-in creates a complete chain of these references from the topmost specified level to the bottom level. This distributed scene graph allows the browser to render only geometry that is visible and at the lowest necessary resolution. This enables a faster rendering time when navigating around the scene.



Rez generates as many levels as the user wants.
In this case Rez starts with 0 and ends with 4.
The 0 Level file references the 1 Level files via a GeoLOD.
The 1 Level files reference the 2 Level files and so on.
This creates the quad tree structure.

Figure 51. Diagram of Rez Distributed Scene Graph, with Files Shown as Boxes

E. SUMMARY

This chapter describes the X3D Geospatial Component and all its Geospatial nodes, with a snapshot of each Editing Panel. It also describes the X3D-Edit support for the Geospatial Component. Finally, it describes the X3D-Earth distributed scene graph generated by Rez.

VI. DETAILED PROBLEM DESCRIPTION AND X3D-EARTH GLOBE ARCHITECTURE

A. INTRODUCTION

The following chapter discusses a detailed description of this thesis and the architecture used to create the X3D-Earth models. The goal of this thesis is to develop an open-source, royalty-free method for generating full-coverage 3D globes using the Extensible 3D (X3D) Graphics International Standard. This thesis utilizes an open-source elevation parsing and image tiling software. The primary means for processing the data is the use of the Hamming Supercomputer. The architecture for this is described in this chapter.

B. APPROACH

1. Using Open Source

The primary purpose of this thesis is to develop a full 3D globe using open source software. The terrain and image processing tool, Rez, developed by Chris Thorne PhD, is an excellent choice for this thesis. It has the capabilities to process multiple input data formats and output multiple output data formats. The code was developed to parse a number of common elevation data formats, i.e., DTED, ASCII, XYZ elevation formats. This is a very useful capability since this thesis is focusing on datasets available to the U.S. Military, and DTED is the primary elevation data format for the U.S. Military. Other open source terrain processing tools were considered, specifically GeoTools, yet the documentation and available examples were not adequate for a quick start to the process. The terrain tile production chain, established by Byounghyun Yoo PhD, was easy to follow and well known at NPS. Thus, Rez was chosen for its versatility and ease of use.

2. Terrain Tile Production Chain

With the tool chosen, the approach is similar to the terrain tile production chain developed by Dr. Yoo, yet on a larger scale.

This process is a good starting point for this thesis. It is well tested and well used at NPS in the Advanced X3D classes. The goal of this thesis is to take the well-established production chain and expand it for use with a dataset for full globe coverage. The contribution of this thesis is to automate the production chain for globe generation or for a large area. The specific architecture for this expanded production chain is covered in the Architecture section of this chapter.

3. Using Hamming Super Computer

The Hamming Super Computer is an excellent resource for utilizing an expanded version of Dr. Yoo's Terrain Tile Production Chain for generating a full 3D globe.

In a memorandum (Jeff Haferman, personal communication, 27 November 2009) on the status of Research Computing at NPS, Dr. Jeff Haferman, Director of ITACS Research Computing, listed the following components for the Hamming Supercomputer:

> The system consists of 144 Sun X6250 blades each with 2 quad-core Intel E5410 processors and 8 GB of RAM for a total of 1152 cores and 1.15 TB of RAM. There is an infiniband interconnect to 32 of the nodes, a 112 TB Sun disk array, and a newly purchased high performance 100 TB DDN disk array connected via infiniband.

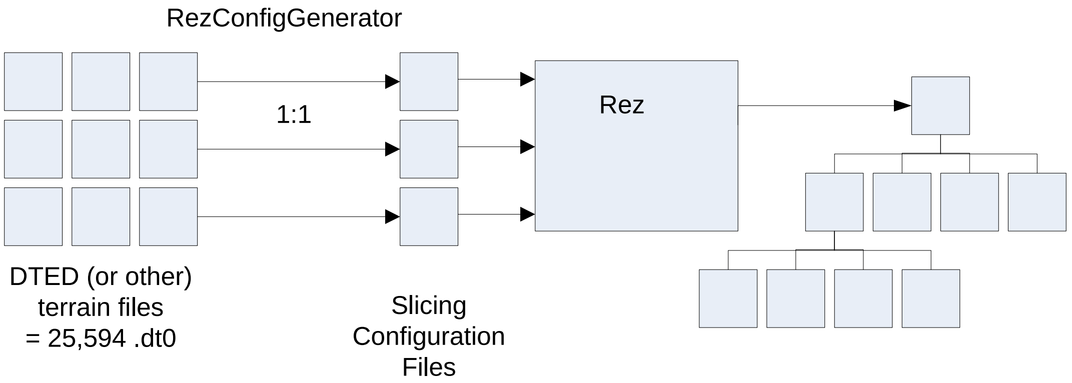
Another source of information for the current capabilities and components for the Hamming Supercomputer is the NPS High Performance Computing Web site, http://www.nps.edu/Technology/HPC/index.html.

The Hamming Supercomputer utilizes a job scheduling utility called Sun Grid Engine (SGE). This scheduling engine is responsible for sorting, tracking, and executing the various jobs that are submitted the Supercomputer. With this engine it is possible to submit large numbers of jobs and each can be executed in parallel with SGE determining which CPU and processors are to be used. This eliminates the need to execute all the jobs from a single CPU and the user does not have to worry about which CPU and processors to use for the jobs. This capability is well suited for the architecture of this thesis. For more information on Sun Grid Engine, go to http://www.oracle.com/us/products/tools/oracle-grid-engine-075549.html.

C. ARCHITECTURE

1. Terrain Tile Generation

The basic architecture for terrain tile generation is illustrated in Figure 52. After the 71 GB of DTED data is transferred to the Hamming Supercomputer file system, the appropriate Processor class is invoked to find all the files and start the process of creating the X3D tile pyramids. First the Processor finds the files and stores the file names for later use. Following this the Processor creates a folder structure that will put a minimal number of X3D tile pyramids within a single directory. The purpose of this is to increase the speed of the display within a browser by decreasing the number of directories to be searched. Next the Processor sends the files, individually, with the appropriate parameters to the RezConfigGenerator class to create the appropriate configuration file. Following the creation of the configuration files, the Processor creates a shell script for each terrain file. Within this shell script includes a Java command-line which launches the Rez code. Within the command-line call the parameters are included, one of which is the newly created configuration file for the respective terrain file. The Processor also creates a text file that contains all the zero level files to be created, for later use when creating the Top Level Globe Scene. Following the creation of all the script files and zero level text file, a master script is created that has a command-line call to the Sun Grid Engine for each terrain script. This process sends all the terrain script files to be scheduled by Sun Grid Engine, which will allocate the appropriate number of processors to execute the individual script files. These are executed in parallel based on the available processors, all of which is automatically organized by Sun Grid Engine. Therefore, each individual terrain file is processed separately within the Rez code. This generates the X3D file pyramids.



Program RezConfigGenerator creates slicing configuration files.

NvtDtedProcessor creates large shell scripts which invokes Rez via Sun Grid Engine (qsub) for all configuration files

Figure 52. Terrain Tile Generation Process

2. Image Tile Generation

The image tile generation is a similar process to the terrain tile generation. The basic architecture is illustrated in Figure 53. The primary difference in the image tile generation is the conversion of the image tiles to a Rez-compatible format and then converting them to dimensions that match the terrain files. This is done in two steps. The first step is to convert the files. In the case of the NaturalVue imagery, the files are originally in GeoTIFF format. Rez is a Java-based program, but it currently does not have the capability to handle GeoTIFF files. Thus the conversion to JPEG is necessary. The ImageConverter class written by Terry Norbraten, Research Associate at NPS, is used for this conversion (Norbraten, 2009). The code is capable of converting one imagery file from a wide array of formats to another file format. The class only converts one file at a time. A similar method to the terrain tile generation of finding the files and creating a shell scripts that launch the ImageConverter class is used. These script files are then launched by a master script that sends the jobs to the Sun Grid Engine of the Hamming Supercomputer. Once these files are converted to JPEG files, they are then processed into sizes that match the terrain files. In the case of the DTED files they are one degree latitude by one degree longitude. Thus the JPEG files must be broken up into this dimension. Originally the JPEG files are in quadsets that cover five degrees latitude by six degrees longitude. Thus the class NvtOneByOneProcessor is used to slice these quadsets into the correct dimensions.

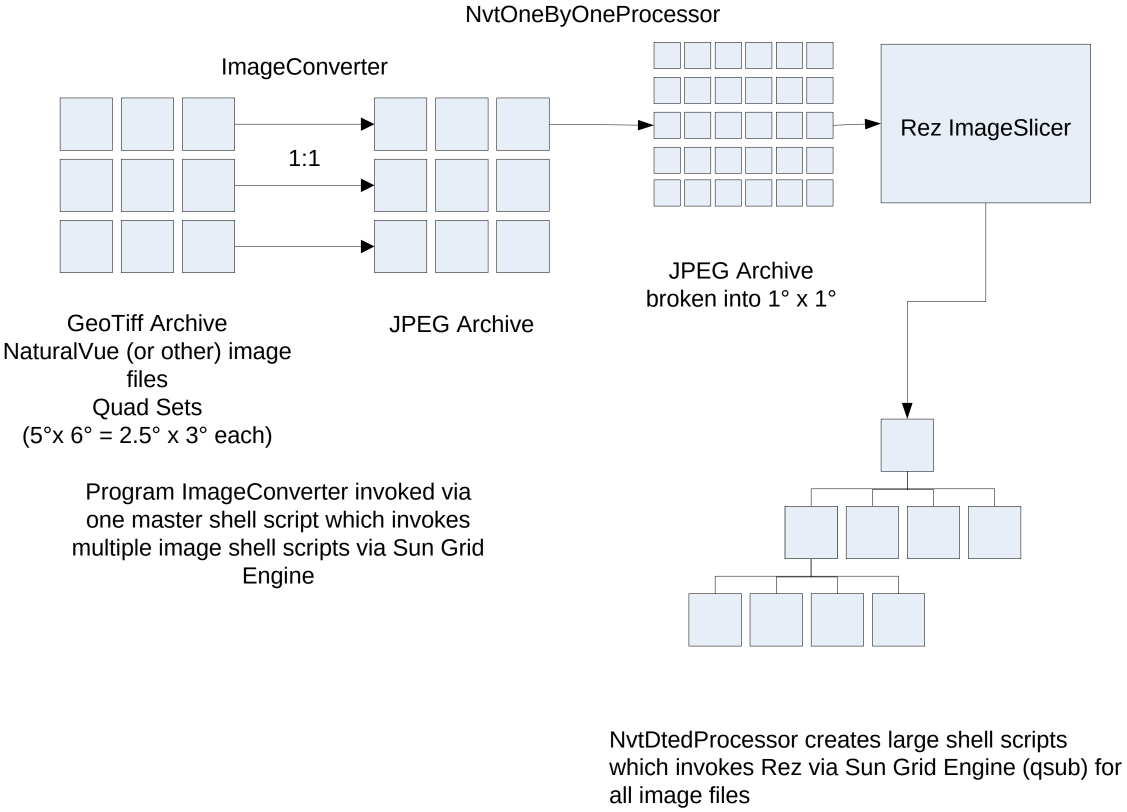
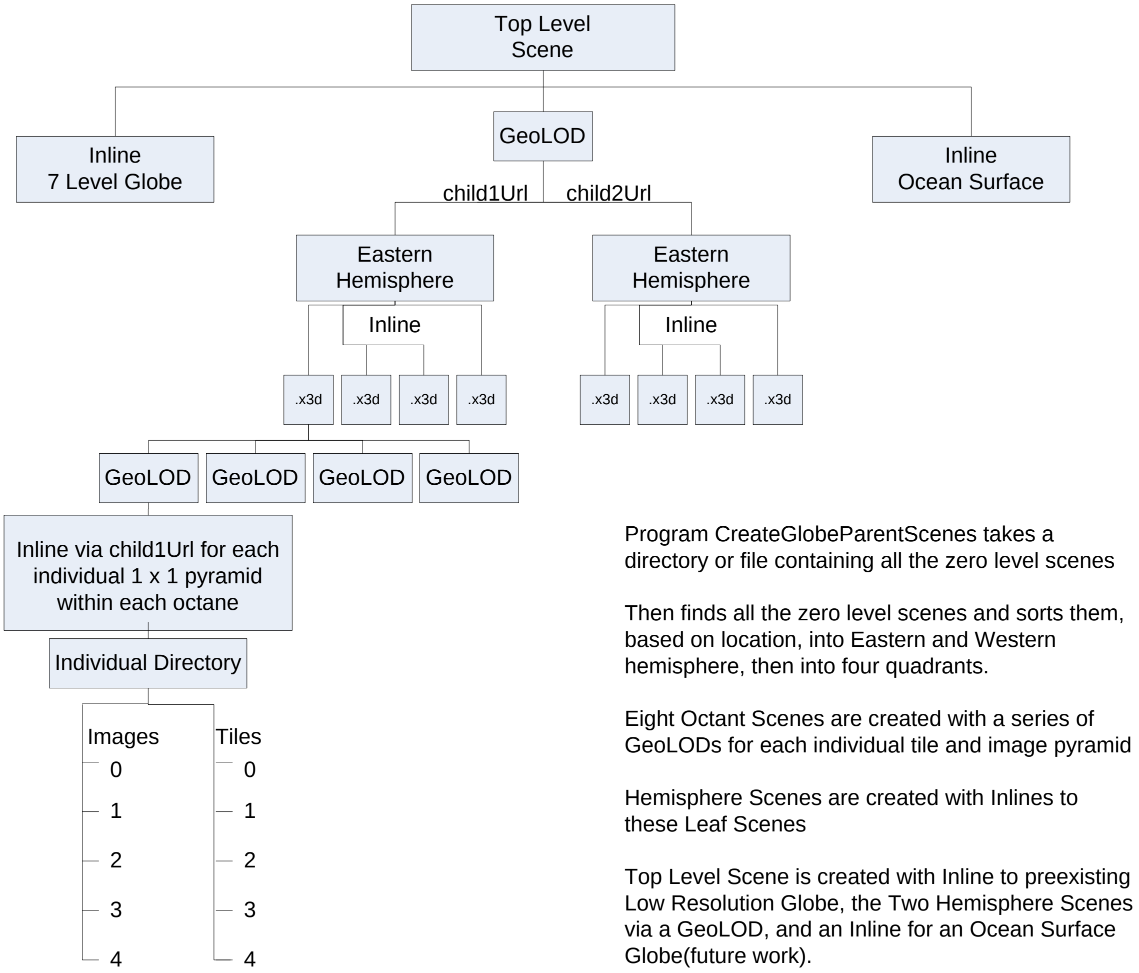

NvtOneByOneProcessor
ImageConverter
1:1
Rez ImageSlicer
JPEG Archive
broken into 1° x 1°
GeoTiff Archive
NaturalVue (or other) image files
Quad Sets
(5°x 6° = 2.5° x 3° each)
JPEG Archive
Program ImageConverter invoked via one master shell script which invokes multiple image shell scripts via Sun Grid Engine
NvtDtedProcessor creates large shell scripts which invokes Rez via Sun Grid Engine (qsub) for all image files

Figure 53. Image Tile Generation Process

3. Top Level Scene Generation

After the terrain tiles and image tiles have been generated, the next step in the process is the generation of the top level scenes. This is required because the tiles are only one degree by one degree pyramids and none of them are connected. The basic architecture of how this is done is found in Figure 54.



Program CreateGlobeParentScenes takes a directory or file containing all the zero level scenes

Then finds all the zero level scenes and sorts them, based on location, into Eastern and Western hemisphere, then into four quadrants.

Eight Octant Scenes are created with a series of GeoLODs for each individual tile and image pyramid

Hemisphere Scenes are created with Inlines to these Leaf Scenes

Top Level Scene is created with Inline to preexisting Low Resolution Globe, the Two Hemisphere Scenes via a GeoLOD, and an Inline for an Ocean Surface Globe(future work).

Figure 54. Top Level Scene Process

The diagram shows the proposed structure of the top-level scenes. These scenes consist of three layers. The program that creates these scenes is the CreateGlobeParentScenes class. The class builds these three levels from the bottom up. It takes all the scenes from the text file that lists all the file names of the zero level files, and it sorts them into eight geographic regions, i.e., Eastern Hemisphere and greater than 90 degrees longitude, or Eastern Hemisphere and less than 90 degrees longitude. These sorted groups will then be referenced into the eight octant scenes via *GeoLOD* nodes and an *Inline* node. These *GeoLOD* nodes enable the multiple scenes to be displayed when within a set range and disable the scenes that are not within range. These scenes are the means of transitioning from the already existing globe scene to the medium-to-high resolution degree by degree scenes. Once the viewer is within the range set in the *GeoLOD,* the degree by degree scenes are loaded and the existing globe scene is no longer loaded.

These eight scenes are referenced by the two Hemisphere scenes via a *GeoLOD* node. These two scenes contain the four scenes that geographically fall within the respective hemisphere. These scenes are used for organizational purposes.

The Top Level Scene is the file that brings all the other scenes together. It has three components to it, the existing globe, *GeoLOD* to the Hemispheres, and the Ocean Surface Globe (possible future work). These three components will represent the full globe. The existing globe is a low resolution globe model, developed prior to this thesis. This model will be the bridge from space to the range that will appropriately load the degree by degree tile sets. The two Hemisphere tiles will be referenced using a GeoLOD node. At a certain range they will be loaded, but not rendered, since the two files only reference other files. This will begin the process of caching the files to be rendered when the viewer gets even closer to the globe. The last component of the Scene is the Ocean Surface Globe. This component is not developed at this time, but could be generated later. Since the existing globe contains the ocean surface and the degree by degree scenes have limited coverage, there is need for another scene that contains a flat solid colored surface to represent the world's oceans. Another possible solution for this is to use bathymetric data for the world's oceans.

This method is not proven in this thesis, yet is proposed as a method for bridging the gap between the two globes.

D. LESSONS LEARNED

1. Initial Architecture

The initial architecture was designed based on the use of the NaturalVue imagery and DTED Level 0 terrain data. Since the imagery is in the form of large JPEG-2000 files that cover five degrees latitude and six degrees longitude, and the use of Java to manipulate JPEG-2000 files is limited and inefficient, it was decided that there would be minimal manipulation of the imagery. The manipulation would involve the terrain files.

In the documentation of the Rez code base, Rez has the capability to group and parse multiple terrain files as one. This capability is a great advantage to the initial architecture; therefore time was devoted to the reading of the imagery. The lessons learned for the simple reading and writing of large image files are in the following section.

Below is the basic architecture of the Initial Terrain Tile Generation process.

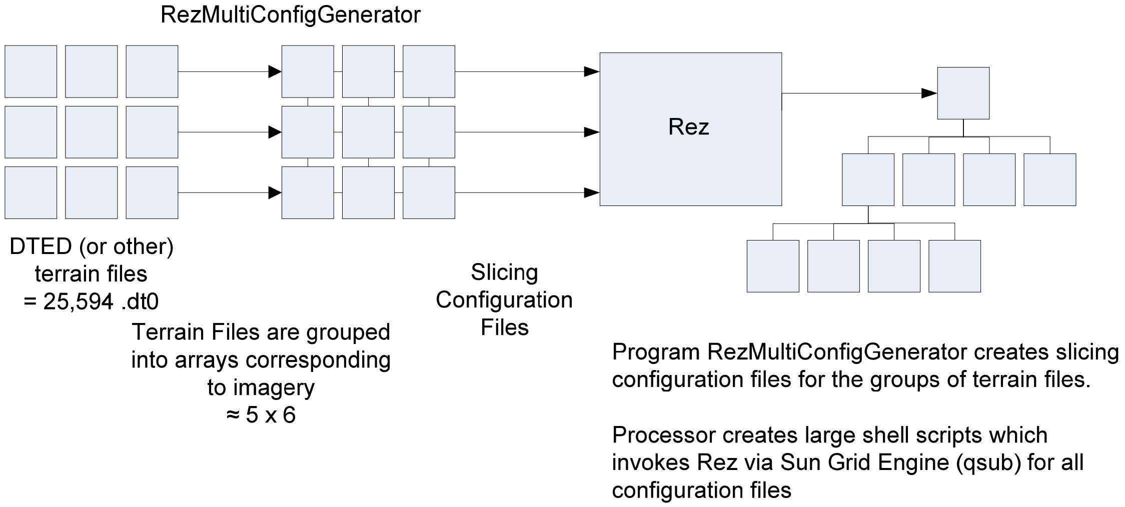

RezMultiConfigGenerator
Rez
DTED (or other) terrain files = 25,594 .dt0
Terrain Files are grouped into arrays corresponding to imagery ≈ 5 x 6
Slicing Configuration Files
Program RezMultiConfigGenerator creates slicing configuration files for the groups of terrain files.
Processor creates large shell scripts which invokes Rez via Sun Grid Engine (qsub) for all configuration files

Figure 55. Initial Terrain Tile Generation Architecture

The DTED files were sorted into groups corresponding to the NaturalVue images. This produced arrays of thirty terrain tiles in five rows of six columns. It was found that, for imagery along the coast or around islands, there were no corresponding terrain tiles for the water. Therefore, the arrays would be filled with null files, causing problems with the Rez code. Thus the architecture described in Section C was developed and used. This explanation of the initial architecture will help the reader to understand why the following issues were encountered.

Figure 56 is the basic architecture of the Initial Image File Generation process.

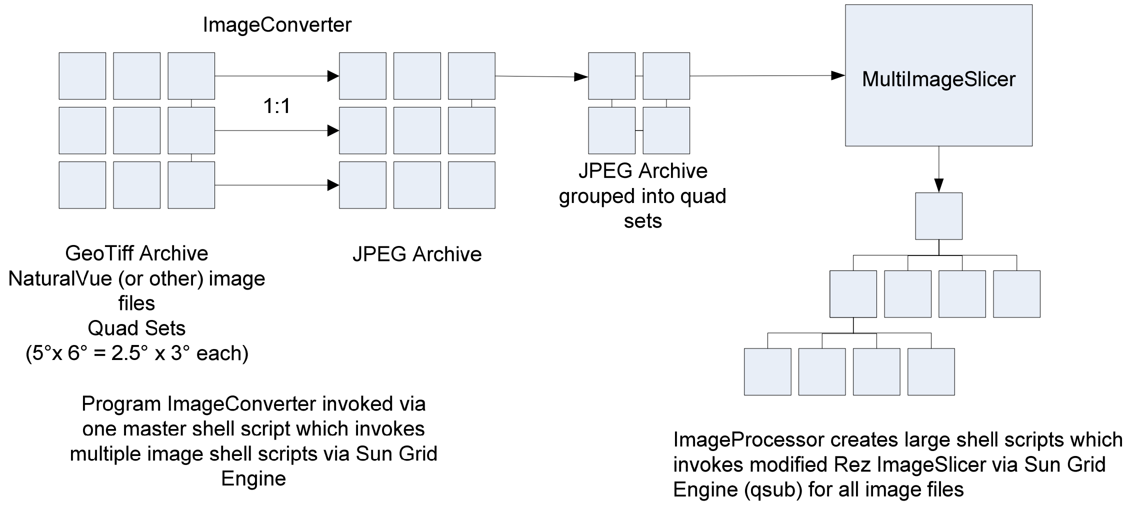

ImageConverter
1:1
MultiImageSlicer
JPEG Archive grouped into quad sets
GeoTiff Archive
NaturalVue (or other) image files
Quad Sets
(5°x 6° = 2.5° x 3° each)
JPEG Archive
Program ImageConverter invoked via one master shell script which invokes multiple image shell scripts via Sun Grid Engine
ImageProcessor creates large shell scripts which invokes modified Rez ImageSlicer via Sun Grid Engine (qsub) for all image files

Figure 56. Initial Image Tile Generation Architecture

The image files were sorted into quad sets and sliced using a class called MultiImageSlicer, developed during this thesis process from Dr. Thorne's basic ImageSlicer class. There were a number of ImageSlicer subclasses developed in this thesis process due to the size of the image files. With both formats of imagery, JPEG-2000 and GeoTIFF, the use of Java was slow and inefficient for directly manipulating the imagery. After discovering the Unix-based tool, ImageMagick, manipulation of the imagery was more simple and efficient. Thus the complications encountered in the terrain manipulation were overcome by switching focus to the imagery. Thus the architecture was changed to the previous discussed architecture in Section C. This explanation of the initial architecture is intended to help the reader to understand why the following issues were encountered.

2. JPEG-2000 Issues

The original imagery received from NGA is in two formats, GeoTIFF and JPEG-2000. Issues with the former are discussed in this section.

JPEG 2000 is a recent development from Joint Photographic Experts Group. The format is "better at compressing images (up to 20 percent plus), it can allow an image to be retained without any distortion or loss" (JPEG, n.d.). After examining the files received from NGA, the size of the images inspected is approximately twenty-five percent of the corresponding JPEG format image. Considering there are over 1000 images that are included for the globe in the archive received from NGA, this is advantageous.

The data stored in these JPEG-2000 files consists of fifteen meter resolution imagery that covers five degrees latitude by six degrees longitude. In this data-preparation process, the imagery needs to be broken up into the imagery pyramids and appropriately named to correspond with the Rez generated terrain tile pyramids. The first step in this process of creating these pyramids is to slice the imagery. The Rez code base contains an image slicer. Thus this image slicer was used as test base for slicing the JPEG-2000 imagery. There were a number of difficulties with the format. Initially the lack of available memory was an issue. The computer being used to run the initial test only had two gigabytes of RAM available to run the Java-based Rez code. The images are on as large as 40,000 by 40,000 pixels, which required more memory to read the images into virtual memory. This issue was resolved by moving the tests to the Hamming Supercomputer, which had a much larger available RAM for use. Yet the images could still not be processed.

After weeks of collaborative efforts with Terry Norbraten, Research Associate at NPS, the cause of these issues was discovered. The primary cause of the issue is the limitations of the Java Advanced Imaging libraries and software for handling the JPEG-2000 files. It seemed to be an issue with the memory management of the JAI library when handling JPEG-2000. The process time duration for this takes an order of magnitude longer when compared to the same process dealing with JPEG images. A conversion to JPEG was a possible alternative solution. A converter was created and utilized, yet the performance issue with the JAI libraries still remained. Furthermore, the GeoTIFF format was utilized for the remainder of the thesis.

3. Java Virtual Memory Issues When Using GeoTIFF files

The second format that is provided by NGA is geographically referenced TIF (GeoTIFF). GeoTIFF are “TIFF files which have geographic (or cartographic) data embedded as tags within the TIFF file” (GeoTIFF FAQ, 2005). The implementation of the GeoTIFF format enabled metadata for geographically referencing imagery possible in an open source non-proprietary format.

The files received from NGA were in quad sets of images in order to have the same coverage as the JPEG-2000 files. Each file covers 2.5 degrees latitude and three degrees longitude. These sets share a common naming convention for each quad set. The convention is that the whole group takes on the latitude and longitude of one of the corners according to the geographic quadrant it is located. Each image in the quad set has this name, followed by an underscore and a corresponding “ul” for the upper left image, or “ll” for lower left, and so on. For example, the images, which cover the area between 35 North Latitude to 40 North Latitude and 84 West Longitude to 90 West Longitude, are named N35-W090_ul, N35-W090_ur, N35-W090_ll, and N35-W090_lr.

a. File Size Issue

The primary issue with the GeoTIFF format was derived from the size of each of the original files. In TIF format, each of the four images of a quad set is in excess of one gigabyte in size. There are approximately 3,000 images that make up the full globe of images, thus the entire image set is 3.4 terabytes of data. This was only a small issue compared to the size of the images in pixels. These large file sizes were due to the uncompressed nature of the TIF format and the metadata attached to each file (including the geographically referencing metadata). Since the images cover such a large area at fifteen meter resolution, the number of pixels can grow to as much as 20,000 by 20,000 pixels per image of the quad set. On another note, the coverage of the images had different dimensions than the terrain files, since they cover two and half degrees latitude. Thus it was concluded that merging the four images into one image would be the best course of action.

Merging the files was an issue. Using the ImageConverter class, as a template, the ImageMergerConverter class was created to merge and/or convert the images into the appropriate coverage and format to use in the Rez ImageSlicer class. This class is very effective with smaller files, but the Java code did not support the massive size of the resulting image when merging the four images of a quad set. In order to merge the images, all four images must be read into memory and them written into new image. The virtual memory available on the Hamming Supercomputer could not handle this task.

b. Slicing One Image at a Time

One solution for this was to write a multiple image slicer, using the Rez ImageSlicer class as a basis. The basic architecture of the ImageSlicer class was constant, yet inside of only reading from one image and splitting it, an algorithm was created to know which quad image was needed to create the next level of detail output image. The use of a quad tree structure for the image pyramids was very useful since the images are already in a quad set. This solution seemed to be the answer, but Java could still not handle reading the four large (1 GB) images into memory. A solution to this issue was to create a class that could handle each image in the quad set independently of the others. This was accomplished in the same manner as the previously mentioned class, yet only one image was loaded at a time. Yet after all this work, the computation was inefficient and it took a large amount of computational time. After exhausting the options using Java, the Unix tool ImageMagick was found as an alternative.

c. ImageMagick as a Solution

ImageMagick is an open-source software package that can be used to create, edit, and compose images. It can read and write more than 100 image formats (Introduction to ImageMagick, n.d.). It is a command-line tool that allows user to manipulate multiple images at once, unlike image processing tools that use a Graphical User Interface. It also has a number of library APIs that interface with many of the common programming languages, such as Magick++ for C++ or JMagick for Java. This software proved to be quite useful in this thesis. The only method for utilizing ImageMagick is via the command-line tools.

ImageMagick demonstrated that it is able to handle extremely large files (i.e., 1 GB) with no issues. Therefore, a new class was created to incorporate the new tool. The process is nearly the same as the previous slicers mentioned before. By utilizing the same loops as in the original ImageSlicer class and using the algorithms for determining which image in the quad set the loop is on, as in the MultiImageSlicer class, only minor changes were needed. However, instead of calling the JAI methods for creating the subsection of the original images and then writing them to file, the new class wrote a Unix command-line call to the operating system ImageMagick to perform the same task. Once these files were complete, they were all executed using the Sun Grid Engine on the Hamming Supercomputer. This method proved to be more efficient, rapid, and effective.

4. Missing Terrain Files

Up to this point in the thesis process, the original production scheme was unchanged with the exception of the classes used to process the imagery. After finding an efficient method for processing the imagery, the terrain files must then be sorted to match the imagery files in coverage. A new class was created to sort the terrain files into arrays which correspond to the imagery. This was achieved and tested for sections of the data in which full coverage was present in the data, i.e., the middle of North America. After continuing the test to include the coastal regions an issue was encountered.

This issue dealt directly with the absence of terrain files to correspond with the imagery for coastal regions or island regions. The solution for this was to create a blank DTED file for those degree by degree sections that were missing. Once this issue was resolved, tests were conducted for processing the terrain files using Rez.

5. Capability of Rez Not Fully Functional

Rez has limited capability for using an array of terrain files to form a single input file. The capability does not function for all cases. It produced multiple errors, even when following the examples and the documentation. These errors occurred in many of the tests with full coverage data, as well as the auto-generated empty DTED files. Along with the issues encountered with the imagery, a change in architecture was necessary.

6. Ultimate Resolution

This section discusses the decision to break up the quad set images into degree by degree files, and how that changed the architecture. Terrain files were sliced as is.

Given the issues discussed in the previous sections and the capability of ImageMagick, a change in architecture was warranted. The handling of large image files in ImageMagick was the key to the change in the architecture. Through this thesis process it was found that the large GeoTIFF files were easily manipulated in ImageMagick within the previous written ImageMagickSlicer class. Therefore, breaking the images into smaller dimensions became possible.

With regards to the manipulation of the terrain within Rez, the easiest and most straightforward process is to generate a pyramid from a single terrain file. The process for doing this for all terrain files in a given directory was already developed. Therefore, reducing the large images to match the dimensions of the terrain files was the major change to the architecture.

Using the ImageMagick command-line tools, the images are manipulated to match the terrain file dimensions. This process is done in the NvtOneByOneProcessor Class, created for this thesis, which changes the dimensions of the images to one degree latitude by one degree longitude. The four quad set images are processed and renamed to match the terrain file dimensions and naming conventions. The DTED file naming convention is based on the latitude and longitude of the terrain represented in the file. The latitude is the file name, and the longitude is the parent folder, i.e., Latitude 30° North – Longitude 030° West, would be w030/n30.dt0. When these files are processed in

Rez, the output .x3d file is w030n30.x3d. Therefore, the images need to be converted to match, i.e., w030n30.jpg. This is important in the new architecture.

7. Producing Millions of Files

For the initial globe generation of this thesis, DTED0 and NaturalVue imagery was used. Each DTED0 file produces an X3D file pyramid to represent that specific location in 3D. In the entire archive of DTED0 files, there are approximately 25,000 files. Each file has data to cover a degree latitude by degree longitude space at one kilometer postings for the elevation data. Within Rez, one of the parameters is the dimensions of each *GeoElevationGrid* node within the X3D files. The dimension is determined by taking the total number of meters in a degree of latitude and longitude. The maximum number is at the equator, in which both latitude and longitude are nominally sixty nautical miles. Each nautical mile is approximately 1852 meters. Thus, a degree of latitude or longitude is 111120 meters. This value is then divided by the resolution of the data, in this case is one kilometer postings, resulting in approximately 111. Then that number is divided by two for each level of detail, in this case five levels. After dividing by two five times we get a value of approximately seven. Rounding up to the nearest ten, the nominal setting for this parameter is ten by ?. This setting is consistent throughout all DTED0 files for this thesis. The dimensions should be varied based on the postings of the input data and the ultimate resolution of the output data.

Once the number of levels is decided, the calculation of number of files per level is based upon the type of pyramid used. For this thesis the quad-tree pyramid is used. Therefore, for each additional level generated the number of files generated increases by a factor of four. Thus there is one file for level zero, four for level one, sixteen for level two and so on. For five levels, zero to four, there are a total of 341 files created per terrain tile pyramid. This number is doubled if there are corresponding image files created. Therefore, the number of files generated grows by a geometric factor of four as the levels and input files increase.

In the case of the initial auto-generated DTED0 and NaturalVue globe, there are approximately 25,000 terrain files and approximately 16,000 image files. With each of these terrain and image files producing 341 files each, each globe creation generates approximately 14,000,000 files. In the testing phase of this thesis, there were two full globes created, as well as a number of other local and regional pyramids. In this manner the number of created files was in excess of 35.8 million files in one work directory. It was found that this was past the limit of the Hamming Supercomputer file system, causing a major crash of the file system.

In an effort to prevent this problem from occurring again, the /HSM/MOVES folder is enabled with a dynamic inode process which allows a much larger number of files. Therefore, all the file production was moved to this folder. Another possible solution is to decrease the number of files generated by combining two consecutive pyramid levels. This can be achieved by adding an additional GeoLOD node containing the GeoElevationGrid of the next consecutive level. This LOD would determine which elevation data is displayed, instead of referencing another file. Additionally, the file would then reference the third level files once the viewer is within a given range on the primary GeoLOD. This structure would continue to the final level. The following is an example of the proposed structure.

```
GeoLOD node
     Contains LOD node and Urls to Level Three Files
     LOD node
          Contains Two Levels
          - Level One GeoElevationGrid
          - Four GeoElevationGrids for Level Two
```

This proposed process decreases the number of files needed for a pyramid of five levels to 69 from 341, including the files for levels zero, one, and three. The levels two and four are included in the level one and three files respectively. This decreases the total number of files in this initial globe to approximately 3,000,000 vice 14,000,000 (a reduction factor of approximately 4).

8. Large Number of Files in a Single Directory

Another characteristic of Rez that is undesirable is that it creates all the output files in a similarly named directory. With the initial architecture, all the output files were deposited into a single directory for each individual level. Therefore, all the level 0 scenes are within one directory, and all the level 1 scenes are in one directory, and so on. This means that for a full coverage globe, there are over 25,000 level 0 scenes within that one directory. This is quadrupled in the level 1 directory, quadrupled again for the level 2 directory, and so on. With any file system, this massive number of files within a single directory drastically affects performance. Thus, a new directory corresponding to the individual terrain file is created. This means that every individual terrain file has its own directory and independent file structure for the X3D scenes and images.

This solves the problem of file numbers within the level directories, but a further modification is needed to limit the number of individual terrain file directories that are contained within a single directory. Additional structure is added above the individual terrain directories that separate the directories into directories that have a maximum of 26 in each individual directory. This is accomplished by producing 1,000 directories each nested in three layers of ten directories. Thus, the number of directories within a single directory is reduced by total number of terrain files divided by 1,000. This enables faster performance when loading X3D scenes and images.

E. SUMMARY

This chapter details the thesis problem and the architecture and processes used. It also describes the lessons learned throughout the process.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. ARCHIVE GENERATION USING SUPERCOMPUTER ASSETS

A. INTRODUCTION

This chapter describes the processes used for archive generation, from obtaining the data, to using the Hamming Supercomputer, and finally exposing the data for external use.

B. OBTAINING ORIGINAL DATA

The primary source of data for this thesis is obtained from NGA. NGA distributes a number of datasets for U.S. government use. These include imagery, terrain data, and cartography. The terrain data includes all three levels of Digital Terrain Elevation Data (DTED), and Shuttle Radar Topography Mission (SRTM) data - which is stored in DTED file format. The imagery data includes NaturalVue imagery, and Controlled Image Base (CIB) imagery. The NaturalVue imagery is fifteen meter resolution, and the CIB imagery is in one meter, five meter, and ten meter resolutions. The cartography data includes Digital Nautical Charts (DNC) and Compressed ARC Digitized Radar Graphics (CADRG).

All datasets can be requested via an electronic form sent to NGA's Custom Media Office. For bulk data needs, like the need for worldwide data, as in the case of this thesis, there is a process for requesting the data via external hard drive or CD/DVD. This request must come from and be mailed to a Department of Defense installation. The electronic form is attached in Appendix A of this thesis. The only cost to the government requestor is the cost of replacing the type of media the data is sent on. In the case of this thesis, the cost of the external hard drives was the only cost for the data, otherwise the data is free. For smaller data requests, NGA has a Web-portal for downloading the data. The Custom Media Office can coordinate access to this site.

For this thesis the data was requested and received in two orders. The first order was for all three levels of DTED, SRTM levels one and two, NaturalVue imagery in JPEG-2000 and GeoTIFF formats, and DNCs. This data was received on three external hard drives totaling in approximately 4.5 terabytes of data. The second order was for CIB imagery in one meter, five meter, and ten meter resolutions, as well as CADRG cartography. This data was received on three external hard drives totaling in approximately five terabytes of data. Replacement drives were purchased and mailed to the Custom Media Office.

C. STORING ORIGINAL DATA

1. MOVES Server

The initial storage of data for this thesis was a server provided by the MOVES Department. This server was mounted by mapping the drive to a laptop. Then the files were “dragged” and “dropped” to the drive. For this case, it was early in the thesis process and the data available was DTED1 and DTED2 on CDs. The fastest way to move the data from this source to the mounted server is to copy the entire disks and paste it to the server drive.

All DTED1 and DTED2 files were transferred in this manner. Once these were transferred, an Ant script was created to transfer the files from the server to the Hamming File System. The following is an example of the task added to the Ant build.xml file within a NetBeans Project:

```
<target name=“moveToHamming” depends=““>
     <scp
          todir=“user@hamming.uc.nps.edu:/work/user/DTED/”
          verbose=“true”  trust = “true”>
          <fileset dir=“Z:/DTED/”/>
     </scp>
</target>
```

The above code used Secure Copy Protocol (SCP) to transfer the data. The todir command is the directory which the data is sent to, and this may require an password, which is illustrated by the asterisks. This task was generated using the NetBeans IDE (can be downloaded at www.netbeans.org). The task was then executed within NetBeans. This type of task is used throughout the thesis for transferring data to the Hamming File System.

2. Hamming File System

Each user of the Hamming Supercomputer is given three folders to use for storing data and executing code. These folders are /home, /work, and /scratch. The /home folder has a smaller capacity than the other two and is used for containing source code and small data sets. The /work and /scratch folders have more capacity and are used for storing larger data sets and processed output files. The difference between the two is that the /scratch folder has a larger size limit, yet it is not backed up since that feature is expensive (both in disk space and time) and since data in that drive is considered transient. The /home and /work folders are both backed up. For this thesis effort, due to the large size of the data sets used, the /scratch folder is used for storing and processing data.

The transferring of data to the Hamming File System is the same process as described in the previous section, yet much of the data was transferred from external hard drive vice mounted drive. Thus one task was executed for each data format on each external hard drive. For example, the task was executed once for the DTED0 files, once for the DTED1 files, once for the NaturalVue JPEG-2000 files, and so on. All of the data sets were transferred to the /work folder for later use.

Ultimately all datasets were transferred to the MOVES folder on Hamming for future use. This enables other users to access the data and there are less limitations on file sizes and no limit on file numbers.

D. GENERATING ARCHIVES FOR DATA

This section describes the process for creating the archives of data within the Thesis code.

1. Automating Shell Script Generation With Ant Scripting

In an effort to minimize user interface with shell scripting, and thus minimize possible errors, an Ant Build script is available for the user. This is created to simplify the process to modifying only one properties file for each individual Demo or Globe Scene created. This properties file is easier for a user to understand when entering the parameters, versus a shell script or series of command line arguments. Once this properties file is modified, a simple command line call is needed to create either the Demo or Globe Scene. A full description of the Properties file and command line calls can be found in Appendix E.

2. Rez Directory Structure

Rez has a specific output directory structure. When Rez is executed it outputs the X3D file pyramid into one folder which it labels /tiles. Within this /tiles folder there are additional directories that correspond with the user determined levels of the pyramid. For example, if the user set the levels to be from zero to two, there would be three directories in the /tiles directory labeled /0, /1, and /2. The directories hold the X3D files of the respective level in the pyramid. The location of the /tiles folder is determined in the configuration file. One of the parameters, stored in the configuration file, is the destination directory. Rez creates the /tiles folder in the destination directory and then populate it with the level directories. If the destination directory is in multiple configuration files, Rez only creates one /tiles folder and only one of each level directory. The X3D files are placed in these directories with the other files of that level from the other source files. Thus the number of files grows with each unique execution of Rez. This can pose a problem as the number of input files grow and the number of levels grows. In the case of this thesis, there are over 25,000 input files with only five levels in the pyramid. At the zero level there would be 25,000 files, yet at level four there would be 6.4 million files. This would create difficulties in accessing these files, due to the number of files the system would have to search to get to a specific file in the directory.

The ImageSlicer class produces a similar directory structure. The primary difference between this and the Rez structure is that it produces an /images folder vice a /tiles folder, and it creates that /images folder in the current working directory. Therefore, it is commonly created wherever the source imagery is located. This is a concern, due to the need to have an archive of imagery separate from the output data. Thus a simple change of directory is needed prior to each execution of the ImageSlicer class. This is achieved within the script files that schedule the execution jobs. The image processor files match the terrain files and image files to ensure they are placed in the correct destination directories. The level directories are create in the /image directory in the same manner as the X3D files in the /tiles directory. The resulting image files are placed in the level directories in the same way as the X3D files. Thus there is a similar issue with accessing and loading the image files.

In an effort to limit the need to search through thousands of files, and in some cases millions, in order to display a location on the globe, a new structure was created. The new structure creates an individual directory for each terrain file, which contains the /tiles and /images directories. This limits the number of files within the level directories. This solves the problem of the massive number of files within a single level directory, yet the globe directory, which contains all the individual terrain file directories, still contains over 25,000 directories. This problem is solved by adding additional structure to limit the number of terrain file directories. For example, three more levels of directories were added to break the individual terrain file directories into groups of 25–26. Thus, there were ten directories with ten directories in each, and an additional directory within the second level, resulting in 1,000 directories for storing the individual terrain file directories. This process is currently hard coded, but could be automated based on the number of files in the data set used.

E. STORING PROCESSED DATA

1. Hamming File System

The generated data is then stored on the Hamming File System. Much of the computational work is done on the /work and /scratch folders. This generated data is then transferred to the /MOVES folder that is connected to the Hamming-fs server to be accessed from the outside.

2. X3D-Earth Server

This section describes the process for transferring and storing the processed data on the X3D-Earth Server.

The X3D-Earth Server is a server that is maintained by the SAVAGE Research Group in the MOVES Institute at NPS in Monterey, California. This server currently has insufficient capacity for handling the entire globe of content, but the smaller regions can be stored on this server. The primary intent of this server is for external access to content, this used for demonstration purposes. The Hamming Supercomputer has an external facing file system that can be mounted to from other external servers. This was the method for displaying the data from an external server. The California Demo can be accessed from http://x3d-earth.nps.edu/California.

F. EXPOSING DATA

1. Hamming-fs

The ability to expose data outside of the Hamming File System is a crucial part of this thesis. The Hamming-fs server is the means for doing this. This server is connected to the MOVES directory within the Hamming File System where all the final demo scenes and globe scenes are stored. From this folder the data is available to be connected to externally facing servers. This server is connected to the X3D-Earth server and able to be accessed. This server can be found at http://x3d-earth.nps.edu. The hamming-fs server is able to be mounted directly to any machine by mapping to hamming-fs.uc.nps.edu. This is achieved by the use of Samba, an open-source network protocol for connecting non-Windows servers to Windows machines (or non-Windows machines).

G. SUMMARY

This chapter describes the process for generating the X3D-Earth archive. It describes the process for obtaining the data, storing the data, preprocessing the data, generating the X3D-Earth scenes, and finally exposing the generated data for external use.

VIII. EXEMPLAR APPLICATIONS

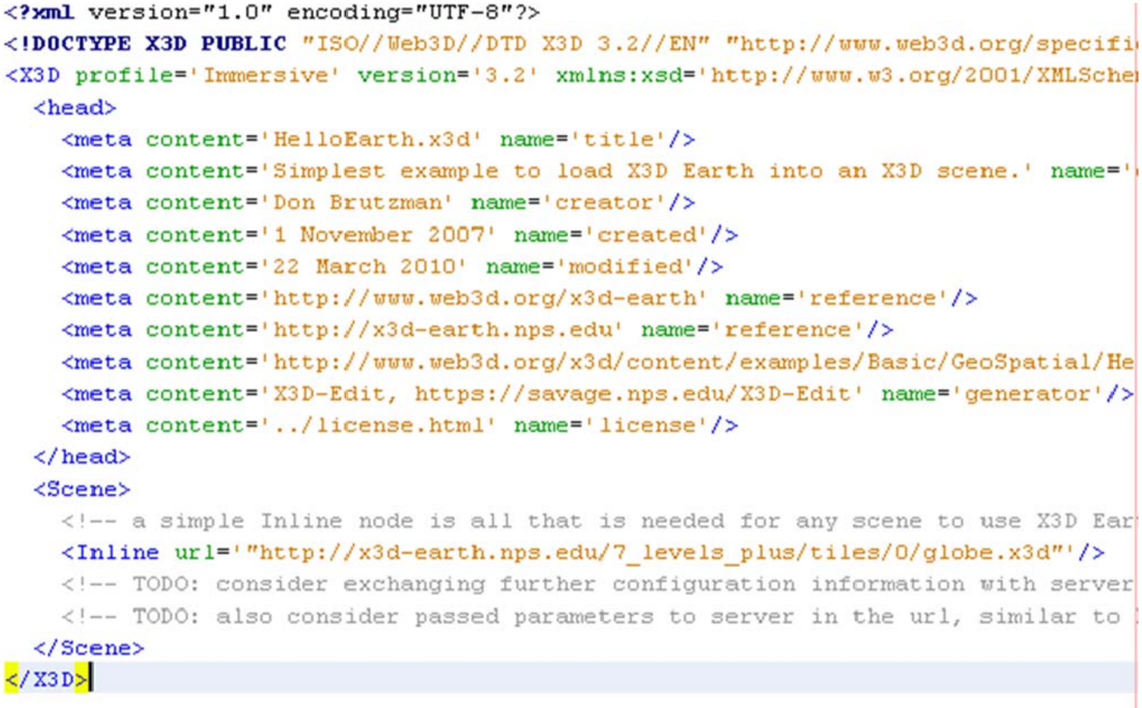
A. INTRODUCTION

This chapter contains descriptions of applications that the results from this thesis can be used for.

B. X3D BASIC EXAMPLES

1. Hello Earth

This model demonstrates the ease of loading an X3D-Earth model into an X3D scene. In the X3D file, the Earth model is simply imported into the file using the *Inline* node. The absolute URL to the globe, in this case http://x3d-earth.nps.edu/7_levels_plus/tiles/0/globe.x3d, is used. For this model, the 7 Levels Plus model developed by Rex Melton is the Earth model used. Figure 57 is a snapshot of the file used to represent this model.



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.2//EN" "http://www.web3d.org/specifi
<X3D profile='Immersive' version='3.2' xmlns:xsd='http://www.w3.org/2001/XMLSche
  <head>
    <meta content='HelloEarth.x3d' name='title'/>
    <meta content='Simplest example to load X3D Earth into an X3D scene.' name='
    <meta content='Don Brutzman' name='creator'/>
    <meta content='1 November 2007' name='created'/>
    <meta content='22 March 2010' name='modified'/>
    <meta content='http://www.web3d.org/x3d-earth' name='reference'/>
    <meta content='http://x3d-earth.nps.edu' name='reference'/>
    <meta content='http://www.web3d.org/x3d/content/examples/Basic/GeoSpatial/He
    <meta content='X3D-Edit, https://savage.nps.edu/X3D-Edit' name='generator'/>
    <meta content='../license.html' name='license'/>
  </head>
  <Scene>
    <!-- a simple Inline node is all that is needed for any scene to use X3D Ear
    <Inline url='"http://x3d-earth.nps.edu/7_levels_plus/tiles/0/globe.x3d"'/>
    <!-- TODO: consider exchanging further configuration information with server
    <!-- TODO: also consider passed parameters to server in the url, similar to
  </Scene>
</X3D>
```

Figure 57. Hello Earth X3D File (From Basic, Geo Spatial: Hello Earth)

Figure 58 is a snapshot of the X3D model, rendered in the Xj3D browser. The 7 Level Plus model is described in more detail in Chapter IX, Section D-1.

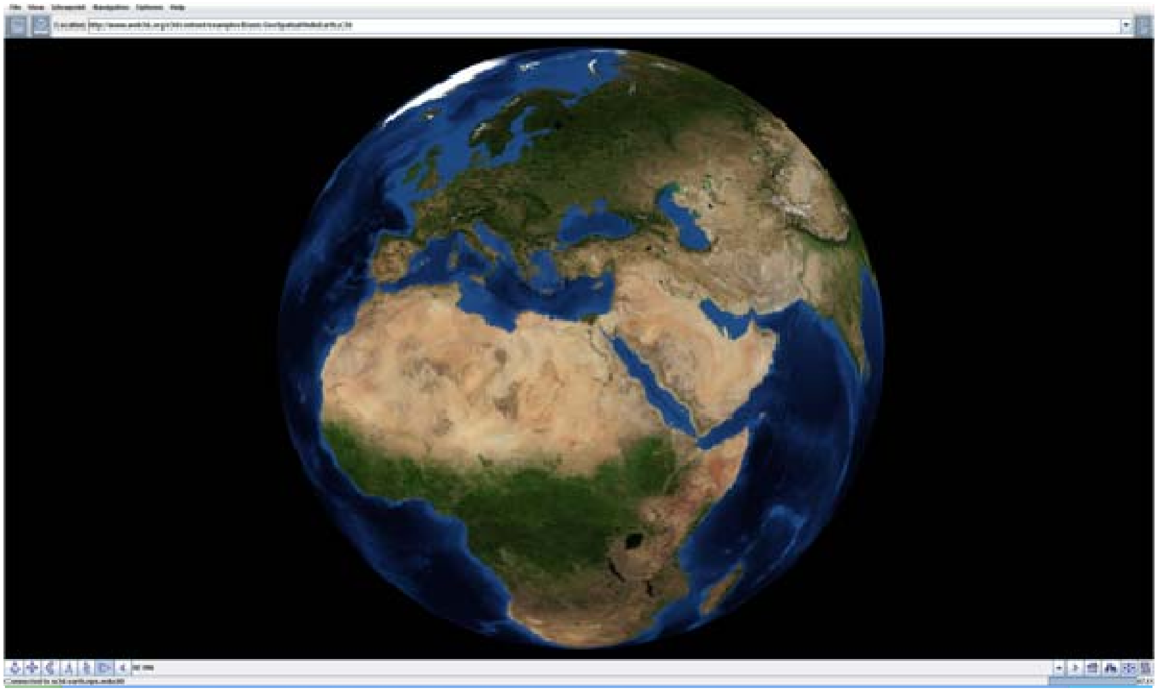
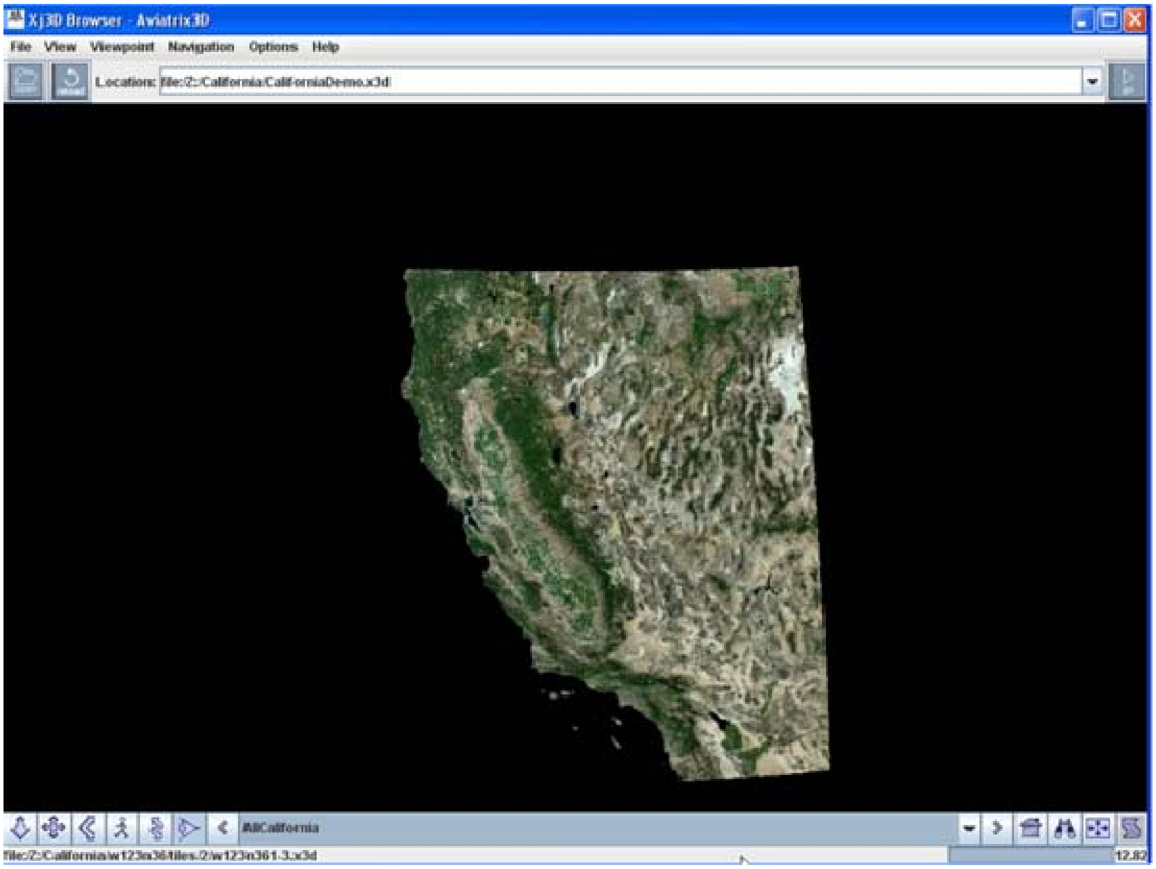


Figure 58. HelloEarth.x3d Model Using BlueMarble Imagery and SRTM 30+ Terrain

2. California Campus Tour

The California Campus Tour is an X3D-Earth Scene generated using the Rez codebase on the Hamming Supercomputer. The data used for the scene is DTED0 for the terrain and NaturalVue for the imagery. The scene is a subset of the globe to illustrate the entirety of the state of California. This demonstration was developed for CENIC in March 2010. This demonstration was generated using the Rez codebase and the RezSubsetByLatLong class to organize and execute Rez on the Hamming Supercomputer.

Apart from the geometry of the scene, it contains a ViewpointSequencer node that iterates through a list of predetermined viewpoints. These viewpoints are of all the Universities and State Universities in California. These latitude and longitude coordinates were programmed into a KML file, by Mike Bailey, a research associate at NPS. Then the KML file was translated into the X3D file by use of an XSLT stylesheet, developed by Dr. Don Brutzman, professor at NPS.



file:Z::California/CaliforniaDemo.x3d

file:Z:/California/w123n36tiles.2/w123n361-3.x3d

Figure 59. California Demo for CENIC 2010

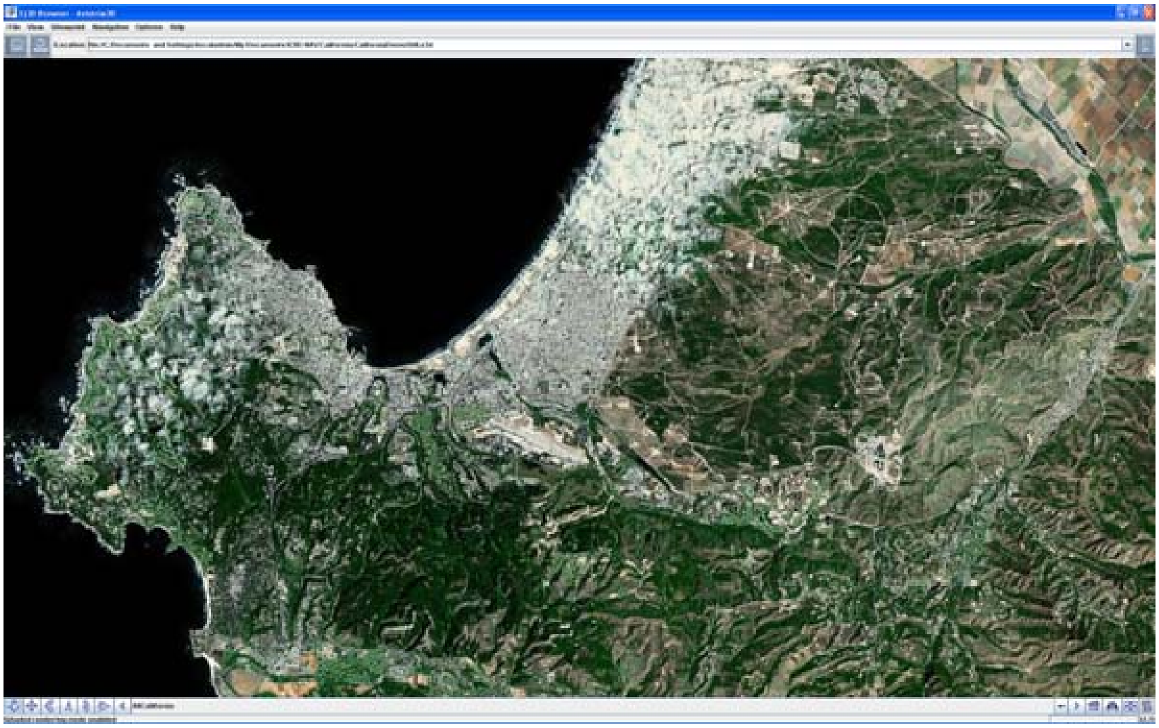


Figure 60. Close Up Screenshot of Monterey in California Demo

C. SUMMARY

This chapter describes some uses for the resulting X3D-Earth models. It gives an example of how an Earth model can be used in an X3D scene.

IX. EXPERIMENTAL RESULTS

This section covers the results for the scenes and globes generated, original file numbers and sizes as well as resulting file numbers and sizes.

A. CALIFORNIA DEMO

1. Original Files

The original files for the California Demo was the DTED Level 0 for the terrain, and the Natural Vue for the imagery. The total number of files and corresponding file size are displayed below in the table.

2. Resulting Files

The resulting files are the .x3d files and the JPEG images, generated by the Rez code. The total number of files and corresponding file sizes are displayed in Table 1.

3. Computational Time

The time reported in the table is the total time it takes for a single CPU to compute all the DTED and image files in minutes (CPU minutes).

Table 1 contains the information about the data used and the resulting terrain and image pyramids.

Data	Number of Files	Size of Files	Computational Time
DTED Level 0	111	4 MB	N/A
NaturalVue 1 x 1	111	2700 MB	N/A
Globe Terrain	37851	65 MB	153.37 minutes *
Globe Imagery	37851	2600 MB	

Table 1. California Campus Tour Statistics

*Time is total CPU time for both imagery and terrain files together.

B. DTED LEVEL 0 AND NATURAL VUE GLOBE

Data	Number of Files	Size of Files	Computational Time
DTED Level 0	25, 594	1.6 GB	N/A
NaturalVue TIF	957	3.4 TB	N/A
NaturalVue JPG	957	218 GB	347.4 minutes
Globe Terrain	8727000	34 GB	35300 minutes
Globe Imagery	6138000	342 GB	

Table 2. DTED0 and NaturalVue Globe

C. METRICS OF INTEREST

The metrics of interest for any globe or subsets created are as follows:

- Number of Files - This is a combined total of image and X3D output file counts.
- Size of Files - This is a combined total of image and X3D output file sizes.
- Computational Time - measured in CPU minutes or amount of time a single CPU would take to create the output. This measurement is based on final output creation.

D. COMPARISON OF AVAILABLE GLOBES

This section compares the available globes to one another. This will include a possible listing of all available globes with corresponding pictures and tables (file numbers, file size, Data Types used, resolution of Terrain and Imagery). The following is an initial list. Information for the first three globes is from the X3D-Earth Web site (X3D-Earth, 2007).

1. Rex Melton's Blue Marble Globe

This globe was created by Rex Melton in October 2007. It was created with JeoSpace software. It is a seven layer globe for global coverage, with an additional five levels for the Boston and Seattle Areas. The terrain data used for this globe is SRTM 30 Plus terrain data at 30 second or one kilometer resolution. The imagery is Blue Marble for the seven layer globe.

2. MBARI 1-Minute Bathymetry

This globe was created by Dr. Byounghyun Yoo in October 2007. It was created with Rez software. It is a seven level globe for global coverage. The data used is derived from the Monterey Bay Aquarium Research Institute bathymetry and terrain data at one minute resolution.

3. SRTM30Plus

This globe was created by Dr. Byounghyun Yoo in October 2007. It was created with Rez software. It is a seven level globe for global coverage, with an addition seven levels for the San Francisco Bay area. The terrain data used for this globe is SRTM 30 Plus terrain data at 30 second or one kilometer resolution. The imagery is Blue Marble for the seven layer globe and LANSAT7 imagery for the San Francisco Bay area.

Globe	Terrain Type	Terrain Resolution	Imagery Type	Imagery Resolution	File Size
Blue Marble	SRTM 30 Plus	1 km	Blue Marble	500 m	126MB
MBARI 1-minute	MBARI Bathymetry	1-minute	Colored Texture MBARI Bath.	1-minute	289MB
SRTM30Plus	SRTM 30 Plus	1 km	Blue Marble	500 m	118MB

Table 3. Globe Comparison Table

THIS PAGE INTENTIONALLY LEFT BLANK

X. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

This thesis pursued ambitious goals. The production of a full-coverage medium to high resolution globe from over 25,000 terrain files and over 18,000 image files is a computationally difficult task. It has proved to be such a task. Even with the use of the Hamming Supercomputer, this task remains difficult. The architecture of process has been proven to be achievable with the creation of the smaller datasets, i.e., California. A full globe was not generated in this thesis process, due to the number of files produced being the limiting factor. With a different distributed scene graph structure that creates fewer files, many difficulties might be overcome.

A significant finding from this thesis is that the sheer number of files produced in this architecture is a limitation. As mentioned above, a method for producing fewer files might be a means of solving this issue, but another is the use of a database. The Hamming Supercomputer has the capability of producing files to a database, yet this method was only discussed in this process. No process was developed to achieve this. Another potential issue for using a database is accessing and reading the produced scenes from an X3D capable browser.

Other data sets can be used with this thesis's architecture. The primary issue with other imagery or terrain data set would be converting the files to have similar naming conventions and dimensions. The example of this thesis is that the imagery and terrain files did not match in these respects. The programs created in the thesis process are used strictly to convert the imagery to match the terrain. If another imagery data set is used, another class will need to be created to convert it to images of the same dimension and naming convention as the DTED files. Likewise if a different terrain file is used, they would need to be modified to match the NaturalVue degree by degree images produced in this thesis. Therefore, once this step is achieved, the programs produced in this thesis are a means to produce globe and subset 3D scenes for use in simulation. This requires the creation of a class that would specifically modify the given data to match.

The best result from this thesis is the ability to create large 3D scenes relatively quickly. This is achieved by the use of the RezSubsetByLatLong class. This class is used to create a 3D scene from DTED and NaturalVue imagery for an area that is represented by a northern and southern latitude boundary and a western and eastern longitude. This capability can be an excellent tool for creating large area scenes for use in a multitude of applications.

B. RECOMMENDATIONS FOR FUTURE WORK

1. Autogenerating Port Scenes from DNCs

Another recommendation for future work consists of using the DNCs. First would be to model various buoys and other common Navaids, using navigationally consistent naming conventions. One such example would be to model a common can-type buoy that is either red or green.

Next step would be to write XSLT stylesheet or other program to convert DNC Navigation aids to KML point positions with key to display buoy type at that specific location.

Next would be to write program to convert bathymetry data in DNC to gridded format and then convert them to degree by degree files to be used in Rez with degree by degree imagery already produced in this thesis.

Then once these terrain and imagery scenes are created overlay the buoys and navigation aids onto the Rez generated scenes. This process could be automated to do any of the DNCs when needed or to produce an archive.

2. Further Optimization of Client Browsers

The only browser used in this thesis is the Xj3D viewer, developed by Yumetech. The primary reason for using this viewer is that it is open-source and able to render X3D-Earth content. For future work, this component of the X3D-Earth process could be optimized to work better over various configurations. This could include finding the right image resolution for the given configuration. This optimization would allow better viewing and increase the usability of the resulting models.

Modify the browser to ensure that there is no duplicate texture loading. Currently Xj3D loads the textures twice. This slows the loading and viewing of scenes. Also, modify the browser to only reload content when forced. This could be a preference item for the user. This would include an option to have only essential content reload. On the other hand, the user should be able to set which content is to be reloaded constantly or rarely. Another improvement is to expand the texture cache to allow pre-caching. This not only improves the loading and viewing of the scenes for demonstration and operation purposes, but it would definitely be ideal for recordings. Currently, Xj3D loads tiles and textures in a right to left and bottom to top fashion. An improvement that would make the loading appear smoother would be to load the files from the center of the current view and then radiating out.

Further improvements include: Turning on a verbose console tracing, enabling the console to write to a debug file versus the console window for large debugging purposes. To improve the interface and functionality a screen overlay to show the viewer's position would be needed. Also, implementing similar interface widgets to Google Earth could be an improvement. Finally, create a coverage map showing the cached textures for trouble shooting and debugging. This could be color coded by coverage depth to show concentration of images layers. This would improve the debugging capability when loading higher resolution geospatial scenes.

3. Further Optimization of Server-Side Auto-Generation

The server-side preprocessing is a key element of this thesis. Further optimization of this process could greatly increase the capabilities of 3D globe generation. The first step is to improve the scene graph. One method was proposed in this thesis, in which two levels are collapsed into a single file, drastically reduced the total number of files produced. Another improvement is to embed and publish a metadata catalog to enable direct transition from higher levels to lower levels without reading intermediate levels.

4. Further Optimization of Network Transmission

The optimization can occur through increased focus on file size versus number-of-files tradeoffs. Another method is to use the existing file compression capabilities of X3D in compressed binary encoding .x3db file format.

5. Bridge the Gap Between Existing Low Resolution Globes and New Higher Resolution Globes

In this thesis process, architecture for making this happen is proposed. Yet this step is not fully achieved in this thesis.

6. Fix Gaps Between Adjacent Tiles

In the resulting scenes of this thesis, there are numerous instances when gaps appear between the various tiles. This is caused by the tiles loading from different levels, i.e., one tile from level 1 and the other from level 2. They have different resolutions in imagery and terrain grids. A way to change this would be to change the LOD range to prevent gaps when closer up. Another way would be to put a background image behind the tiles, so that these gaps would be less noticeable.

7. Develop Efficient and Adaptable Lighting Scheme

Currently, six point lights are created to illuminate any model on the globe, including the terrain. This was added to enhance the default headlight, which was not powerful enough to illuminate the scene at all angles. A more efficient method would be to have a light that follows the viewer, essentially a much more powerful headlight. This could be done utilizing the *GeoProximitySensor* node.

8. Fix Limitation with GeoLOD Node

Currently the GeoLOD node only contains a single root node geometry, and only supports higher-resolution children through a URL reference. In order to embed the four children nodes directly into the file, the GeoLOD node would need this capability. This new capability would allow further optimization of the files generated in the server-side preprocessing stage, which would create fewer total files.

9. Expose Outputs Through NPS Firewall

This is needed to expose the data generated by this thesis. The FOUO data could be exposed on the nps.navy.mil server, while the open-access data could be exposed on the nps.edu server.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A: THESIS SOURCE CODE

All source code for this thesis can be found on the Source Forge Web site under the X3D project. Within this project it is under tools/x3d-earth/GlobeGenerator. The URL is http://x3d.svn.sourceforge.net/viewvc/x3d/www.web3d.org/x3d/tools/x3d-earth/GlobeGenerator/.

The code is organized into two Source folders under the folder /src. These two packages are GlobeGenerator and ImageSlicer. The GlobeGenerator package contains all the terrain processing and scene generating code, while ImageSlicer contains the image preprocessing code.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B: DATA REQUEST FORM

This appendix contains the form for requesting terrain, imagery, and cartography from NGA.

A. CUSTOM MEDIA REQUEST FORM V7

The form can be found at the following link:

https://sourceforge.net/projects/tourtelotte/files.

Information required for the form is the requestor's name, phone number, e-mail, shipping address, media preference (Hard Drives or CD/DVD), the requested coverage, i.e., worldwide, and the requested products. This form also includes a description of the file sizes of the products available.

Once the form is completed, e-mail it to CustomMedia@nga.mil or fax it to 636-321-5146.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C: REZ CONFIGURATION FILE

This appendix contains an example of the Rez Configuration File.

A. REZ CONFIGURATION FILE

This file is an essential piece to running the Rez code. It contains a number of input parameters that generally remain constant, thus they are included in the configuration file vice the command-line parameters. On the following page you find a complete example of the Rez Configuration file, generally a .txt file. This file can be copied into a .txt file and modified for use with Rez.

1. Format of Configuration File

The format of the configuration file is based on that of a MSDOS batch file. The '#' character at the head of the line signifies a commented line, thus the program does not read that line. It is meant only for the user, as instruction. The other lines without the '#' character are meant for the program. The first string in each of these lines signifies which parameter is to be set, and the following string is the value of the parameter. In some cases, this parameter can be blank, as in the case of the GUI and Info parameters. If there is no need for the GUI, i.e., the command-line is the preferred method of using Rez, then the line would contain "GUI " and no following string for the parameter. This tells Rez that there is no need for a GUI. This is similar for the Info parameter.

2. Further Documentation

Further documentation can be found at http://www.rez3d.com/Rez.pdf. This is the official documentation for Rez.

```
# Configuration file for Rez.
# This file tells Rez which plug-in classes to use for parsing input,
# generating a multirez tree of tiles, the user input GUI and the main
# “scene” output.
# It also specifies the directory for text properties which can be used
# to alter the tile output without having to alter code.
#
# Source directory where the grid data is (use “./” for current dir)
sourceDirectory ./
#
# Destination directory to put the final tree
destinationDirectory ./
#
# The main scene creation plug-in
Scene rez.plugins.scene.X3DScene
#
# The GUI plug-in (requires an extra button push, but good to review
# arguments given to Rez)
# GUI rez.plugins.gui.SimpleGUI
GUI
#
# The number of separate, but adjacent tile arrays which should join
# seemlessly
TileArrays 1
#
# The dimensions of the next tile array
TileXDim 1
TileYDim 1
#
# The max number of digits after decimal point for height values in text
# output (like VRML)
MaxDecimalPrecision 0
#
# Does each successive row in the output tiles take one from North to
# South? It does for for Standard ElevationGrids, but GeoElevationGrids
# go from South to North
NorthToSouthRows n
#
# Does each successive row in the input tiles take one from North to #South? Per MIL-PRF-89020B,
DTED Levels 0-2 rows go from South to North #and this is fully accounted for in the DTEDParser
sourceDataNorthToSouthRows y
#
# Additional Info on the data that may be needed by parser
Info ....
#
# The elevation parser class
Parser rez.plugins.ParseDTED
#
# The tile builder class
Tiler rez.plugins.geoX3d.GeoX3DTile
#
# One row in a tile array
TileList W123N45.dt0
```

APPENDIX D: IMAGE PREPROCESSING

This appendix contains a complete description of how to generate and modify a properties file for execution of the GlobeGenerator Ant Script.

A. CREATING THE PROPERTIES FILE

The ImageProcessor.xml file for executing the Ant Script is located at /HSM/MOVES/ or can be obtained at https://sourceforge.net/projects/tourtelotte/files.

If not executing from the /HSM/MOVES/ folder, download the .jar files from https://sourceforge.net/projects/tourtelotte/files/Jars.zip. Place these .jar files in the same directory. Also, locate the terrain and imagery files to be used. These files must be of the same naming convention as the standard DTED files, i.e., /w111/n11.dt0 for terrain and w111n11.jpg for imagery.

Below is a description of each parameter in the Properties file.

1. Parameters

The parameters for the MultiImageConverter class are listed in Table 61.

Parameter	Description
SourceDirectory	This is the absolute file name of the directory of the imagery to be converted, i.e., c:/Images/OldImagery/
DestinationDirectory	This is the absolute file name of the directory that the user wishes the converted imagery to be stored, i.e., c:/Images/NewImagery/
NewExtension	The extension of the new file format
JarFolder	The absolute location of the folder containing the Jar files used for processing.
ImageExtensions	This is the original extension of the file format. If used for OneByOneProcessor, this is the extension of the images to be processed into One degree by One degree
Rows	Number of Degrees Latitude in the original images
Columns	Number of Degrees Longitude in the original images

Figure 61. Parameters for the MultiImageConverter Class

2. Process

A blank Properties file can be obtained from https://sourceforge.net/projects/tourtelotte/files or can be cut and pasted from Section B. Once this file is obtained or created, fill in the appropriate information. Ensure that all file names are either in the current working directory, i.e., where the ImageProcessor.xml file is located, or include absolute pathnames.

After the Properties file is complete, execute the following command line calls for the given task:

To convert the images from one format to another:

```
ant -buildfile ImageProcessor.xml -propertyfile PropertyFile.txt MultiConverter
```

To create the One Degree by One Degree images (all on one line):

```
ant -buildfile ImageProcessor.xml -propertyfile PropertyFile.txt
NvtOneByOneProcesser
```

B. EXAMPLE PROPERTY FILE

```
SourceDirectory = /HSM/MOVES/OriginalImageryFolder

ImageExtension = tif

DestinationDirectory = /HSM/MOVES/OutputImageFolder

JarFolder = /HSM/MOVES/Jars

#This is used for Image Conversion Only

NewExtension = jpg

#Rows and Columns are only used for creating One by One Images

Rows = 5

Columns = 6
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX E: TERRAIN TILE AND SCENE GENERATION

This appendix contains a complete description of how to generate and modify a properties file for execution of the GlobeGenerator Ant Script.

A. CREATING THE PROPERTIES FILE

The build.xml file for executing the Ant Script is located at http://x3d.svn.sourceforge.net/viewvc/x3d/www.web3d.org/x3d/tools/x3d-earth/GlobeGenerator/.

If not executing from the /HSM/MOVES/ folder, download the .jar files from https://sourceforge.net/projects/tourtelotte/files/Jars.zip. Place these .jar files in the same directory. Also, locate the terrain and imagery files to be used. These files must be of the same naming convention as the standard DTED files, i.e., /w111/n11.dt0 for terrain and w111n11.jpg for imagery.

Below is a description of each parameter in the Properties file.

1. Parameters

The parameters for the Property file are listed in Figure 62.

Parameter	Description
TerrainDirectory	This is the absolute file name of the directory that contains the terrain data
TerrainExtension	The extension of the terrain files, to be used by the FileFinder class.
ImageDirectory	This is the absolute file name of the directory of the imagery to be converted, i.e., c:/Images/OldImagery/
ImageExtension	The extension of the image files, to be used by the FileFinder class.
DestinationDirectory	This is the absolute file name for the directory in which the new imagery is stored.
JarFolder	A string containing the absolute location of the Jar files used for processing the imagery.
MasterConfigurationFile	This is the master configuration file that contains many of the settings that go in the configuration files.
FirstLevel, LastLevel	These are the first level and last level of the X3D and image pyramids.
TerrainResolution	The single value resolution of the GeoElevationGrid in the created X3D scenes, i.e., 20 entered, results in a 20 x 20 GeoElevationGrid.
ImageResolution	The single value resolution of the images, this number is the height and the width, i.e., 256 entered, results in a 256 x 256 image.
NorthernBoundary, SouthernBoundary	Northern and Southern most latitudinal boundaries of the Demo Scene. Must be in the following form: n11 or s11. Not used if creating the globe.
EasternBoundary, WesternBoundary	Eastern and Western most longitudinal boundaries of the Demo Scene. Must be in the following form: e011 or w011. Not used if creating the globe.
World	Absolute pathname to a preprocessed low resolution globe

Figure 62. Parameters for the DtedNvtProcessor Class

2. Process

A blank Properties file can be obtained from https://sourceforge.net/projects/tourtelotte/files or can be cut and pasted from Section B. Once this file is obtained or created, fill in the appropriate information. Ensure that all file names are either in the current working directory, i.e., where the build.xml file is located, or include absolute pathnames.

After the Properties file is complete, execute the following command line calls for the given task:

To create the X3D and image file pyramids for the Globe:

```
ant -propertyfile PropertyFile.txt DtedNvtProcessor
```

To create the Parent Scenes for the Globe:

```
ant -propertyfile PropertyFile.txt CreateGlobeScene
```

To create the X3D and image file pyramids for a Demo/Subset of the Globe:

```
ant -propertyfile PropertyFile.txt RunSubSet
```

To create the Parent Scenes for the created Demo Scene:

```
ant -propertyfile PropertyFile.txt CreateDemoScene
```

These calls could be done in pairs, i.e., when creating a Demo Scene:

```
Ant -propertyfile PropertyFile.txt RunSubSet CreateDemoScene
```

Ensure that the scene creator follows the pyramid generator.

B. EXAMPLE PROPERTY FILE

```
TerrainDirectory = /HSM/MOVES/DTED/dted0/

TerrainExtension = dt0

ImageDirectory = /HSM/MOVES/NaturalVueJpg1x1

ImageExtension = jpg

DestinationDirectory = /HSM/MOVES/NewDemo

JarFolder = /HSM/MOVES/Jars

MasterConfigurationFile = /HSM/MOVES/DTED/masterConfigFile.txt

FirstLevel = 0

LastLevel = 0

TerrainResolution = 10

ImageResolution = 256

NorthernBoundary = s34

SouthernBoundary = s35

EasternBoundary = e18

WesternBoundary = e18

World = /HSM/MOVES/SRTM30Plus/world.x3d
```

APPENDIX F: CHECKING OUT SOURCE CODE FROM SOURCEFORGE

The location of the source code is http://x3d.svn.sourceforge.net/viewvc/x3d/www.web3d.org/x3d/tools/x3d-earth/GlobeGenerator/. The code can be checked out using Tortoise SVN or NetBeans. Since the code is developed as a NetBeans Project, that method is described below.

First, open NetBeans and select “Team” from the top menu. Then select “Subversion” and finally “Checkout.”

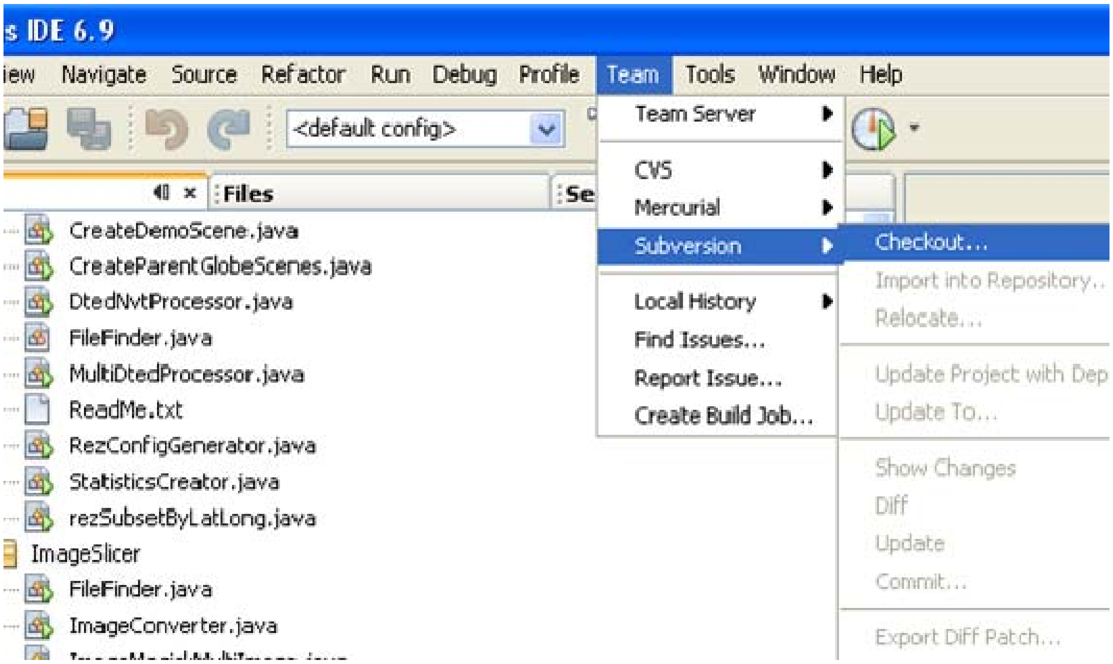
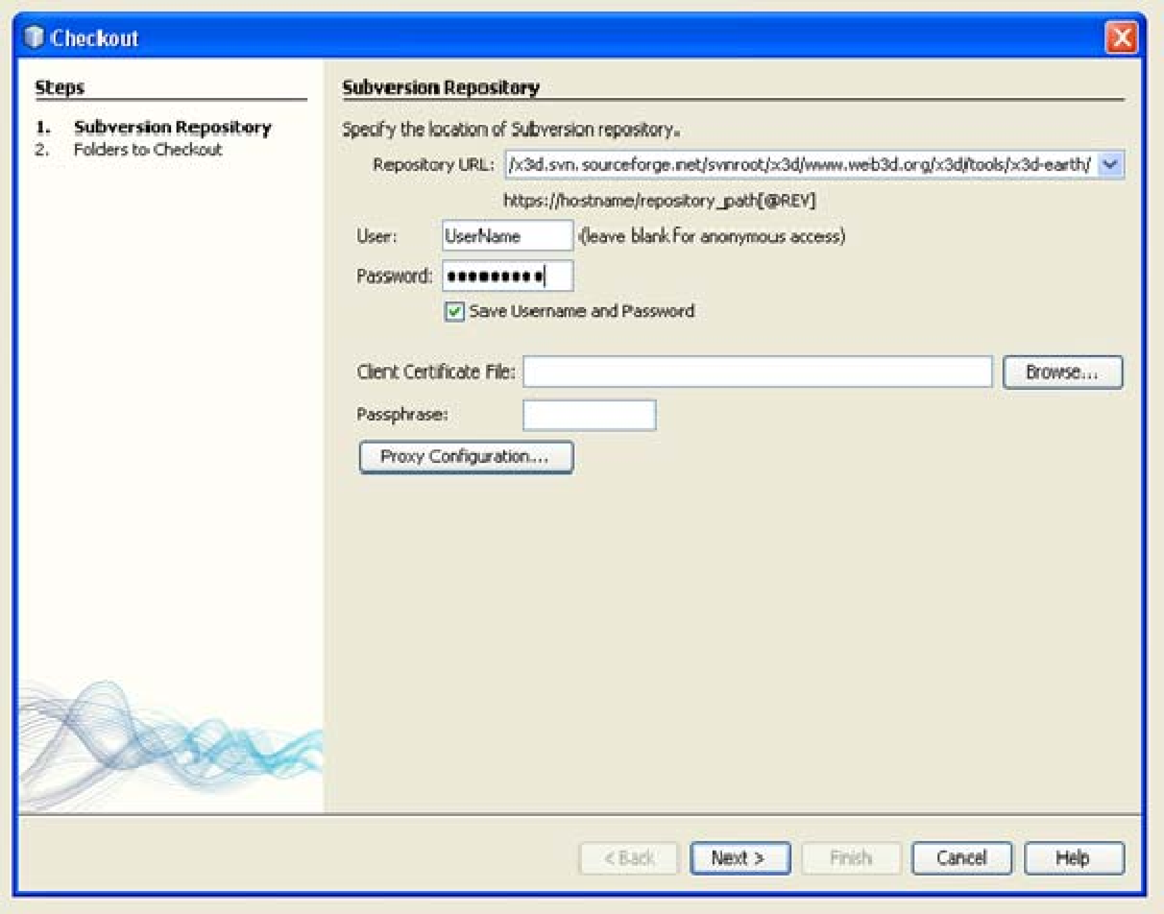


Figure 63. Subversion Checkout in Netbeans

The following panel will be a window to input the URL location of the Subversion Repository. In this window enter: https://x3d.svn.sourceforge.net/svnroot/x3d//www.web3d.org/x3d/tools/x3d-earth into the Repository URL place. Then a User Name and password are needed. Then select next. Figure 64 shows the panel.



Subversion Repository

Specify the location of Subversion repository.

Repository URL: /x3d.svn.sourceforge.net/svnroot/x3d/www.web3d.org/x3d/tools/x3d-earth/

https://hostname/repository_path[@REV]

User: UserName (leave blank For anonymous access)

Password: ●●●●●●●●●

☑ Save Username and Password

Client Certificate File:

Passphrase:

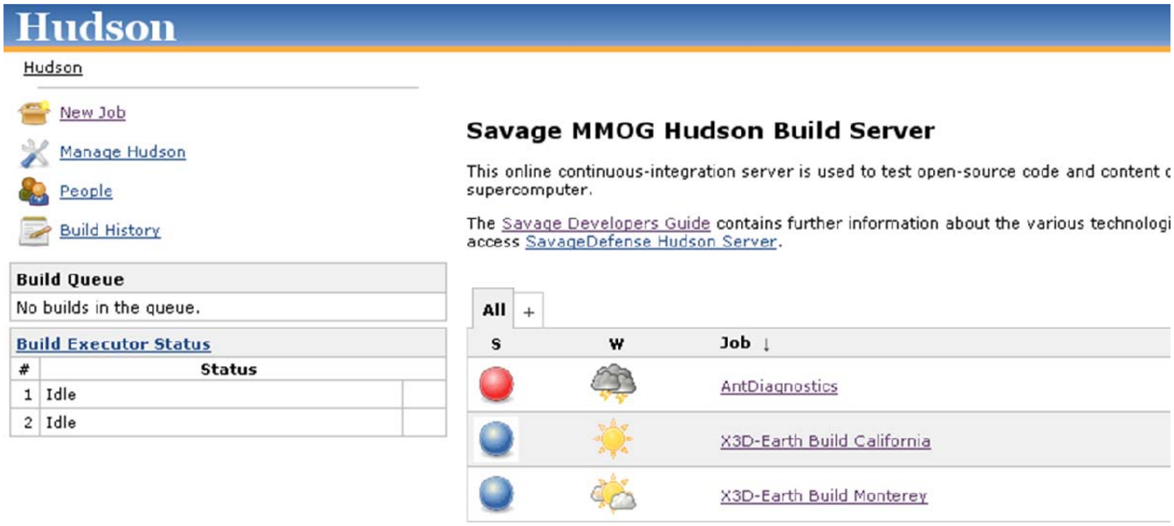
Figure 64. Subversion Repository

Next, a local folder location for the checkout project will be needed. Once this is filled in, select Finish and the project is checked out.

APPENDIX G: HUDSON BUILD TASKS

The SAVAGE MMOG Hudson Build Server is an “online continuous-integration server is used to test open-source code and content distributions” (SAVAGE, 2010). Located at http://mmog.ern.nps.edu/hudson/, this particular server is restricted to NPS only. Two additional corresponding servers are available for unclassified open access and unclassified restricted access for Official Use Only (FOUO). The current build projects are two tasks for this thesis, “X3D-Earth Build Monterey” and “X3D-Earth Build California,” The first builds a Demo Scene of Monterey, CA, and the second builds a Demo Scene of California. These Hudson tasks execute Ant build scripts executing on the NPS Hamming Supercomputer.

For further documentation on the Hudson build server at NPS, go to https://savage.nps.edu/developers.html#Hudson.



Savage MMOG Hudson Build Server

This online continuous-integration server is used to test open-source code and content c
supercomputer.

The Savage Developers Guide contains further information about the various technologi
access SavageDefense Hudson Server.

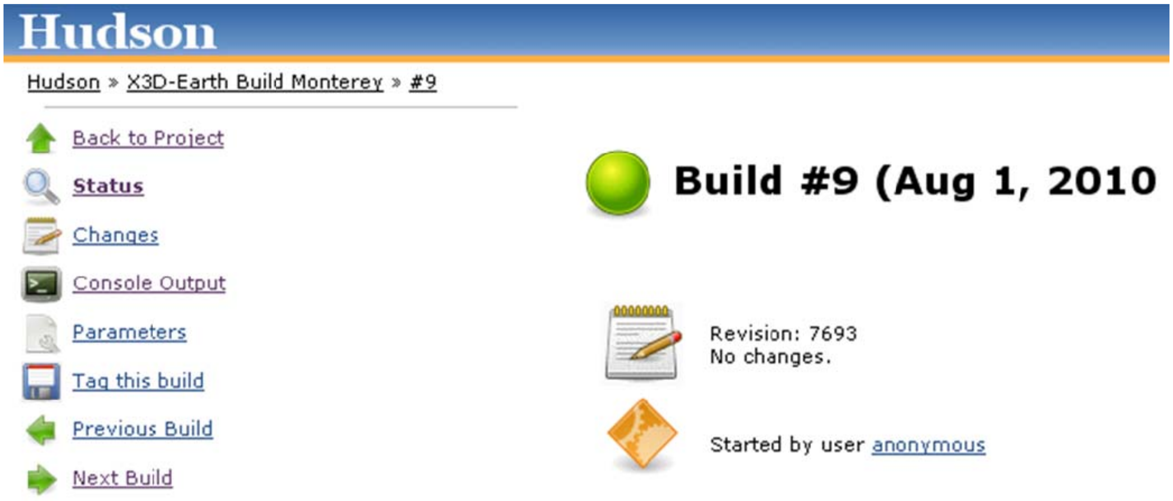
Build Queue
No builds in the queue.

#	Build Executor Status Status
1	Idle
2	Idle

S	W	Job ↓
●	☁	AntDiagnostics
●	☀	X3D-Earth Build California
●	⛅	X3D-Earth Build Monterey

Figure 65. Hudson Dashboard

To execute the Builds, click on the appropriate task in the window. Once the task is selected the next window will show the Build History as well as a number of other links to modify the Build. To execute the Build, click Build Now. Then click the “Build” Button to run the task. Following this, in the Build History window, the new task will be created. Click on that new task and it will provide you with a new window to monitor the task. The following figure shows a snapshot of this window.



Build #9 (Aug 1, 2010

Revision: 7693
No changes.

Started by user anonymous

Figure 66. Hudson Build Window

Note that this process is identical to invoking project tasks on a personal computer. The same Ant build tasks can be run via command line or from within an IDE like NetBeans. This third option of running via a Hudson server provides independent and regularly recurring repeatability of build task, thus ensuring code base stability throughout the lifecycle development by programming teams in different locations.

APPENDIX H: GOOGLE EARTH HISTORY

This section comes from the Google Corporate Milestones available at http://www.google.com/corporate/history.html.

- October 2004, “We acquire Keyhole, a digital mapping company whose technology will later become Google Earth.”
- June 2005, “We unveil Google Earth: a satellite imagery-based mapping service combining 3D buildings and terrain with mapping capabilities and Google search.”
- September 2005, “Overlays in Google Earth illuminate the devastation wrought by Hurricane Katrina around New Orleans and the Gulf Coast. Some rescue teams use these tools to locate stranded victims.”
- September 2006, “Featured Content for Google Earth includes overlays from the UN Environmental Program, Discovery Networks, the Jane Goodall Institute, and the National Park Service.”
- June 2007, “Featured Content for Google Earth includes overlays from the UN Environmental Program, Discovery Networks, the Jane Goodall Institute, and the National Park Service.”
- August 2007, “Sky launches inside Google Earth, including layers for constellation information and virtual tours of galaxies.”
- April 2008, “A new version of Google Earth launches, incorporating Street View and 12 more languages. At the same time, KML 2.2, which began as the Google Earth file format, is accepted as an official Open Geospatial Consortium standard.”
- May 2008, “Following both the Sichuan earthquake in China and Cyclone Nargis in Myanmar (Burma), Google Earth adds new satellite information for the region(s) to help recovery efforts.”
- October 2008, “We introduce Google Earth for the iPhone and iPod touch, complete with photos, geo-located Wikipedia articles, and the ability to tilt your phone to view 3D terrain.”
- February 2009, the latest version of Google Earth makes a splash with Ocean, a new feature that provides a 3D look at the ocean floor and information about one of the world's greatest natural resources.”
- July 2009, “We launch Moon in Google Earth on the 40th anniversary of the moon landing. The tool features lunar imagery, information about the Apollo landing sites, panoramic images shot by the Apollo astronauts and narrated tours.”

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Aeroplanner. (n.d.). "What is CADRG." Retrieved 19 May 2010 from http://www.aeroplanner.com/cadrg/what.cfm

Bacharach, S. (2004). "The OGC – A unique organization offering unique benefits." Open Geospatial Consortium White Paper from OGC official site, http://www.opengeospatial.org/pressroom/papers

Blais, C. & Norbraten, T. (n.d.). "SAVAGE Metadata Analysis Language." Retrieved 9 July 2010 from SAVAGE official site. https://savage.nps.edu/Savage/Tools/SMAL/SmalOverview.ppt

BING. (2010a). "Add innovation apps to your site." Retrieved 9 July 2010 from BING Maps Platform official site. http://www.microsoft.com/maps/developers/web.aspx

BING. (2010b). "Licensing." Retrieved 9 July 2010 from BING Maps Platform official site. http://www.microsoft.com/maps/product/licensing.aspx

Brutzman, D., Sadagic, A., & Norbraten, T. (2007) "Extensible 3D (X3D) Earth technical requirements workshop summary report." From Web3D official site. http://www.web3d.org/x3d-earth/workshop2006/X3dEarthTechRequirementsWorkshopNovember2006.pdf

Brutzman, D. (2010). "X3D Schematron validation and quality assurance." Retrieved 9 July 2010 from Web3D official site. http://www.web3d.org/x3d/tools/schematron/X3dSchematron.html

Coldwell Banker Commercial. (2010). " Map it." Retrieved 13 May 2010 from Coldwell Banker Commercial official site. http://www.cbcworldwide.com/enhanced/index.php?what=&type=property&subType=map&where=#t=0

Digital Terrain Elevation Data. (2008). Retrieved 7 April 2010 from National Geospatial-Intelligence Agency official site. https://www1.nga.mil/ProductsServices/TopographicalTerrestrial/DigitalTerrainElevationData/Pages/default.aspx

Digital Earth. (2002). "TsmAPi Library." Retrieved 13 May 2010 from Digital Earth official site. http://www.ai.sri.com/tsmApi

Doyle, A. & Reed C. (2001). "Introduction to OGC web services." Open Geospatial Consortium White Paper from OGC official site. http://www.opengeospatial.org/pressroom/papers

Dykstra, J. (n.d.) "NaturalVue-Rev2: A Global Landsat Image Dataset." Rockville, Maryland: MDA Federal Inc. from http://www.mdafederal.com/digital-imaging/earthsat-naturalvue/NaturalVueWhitePaper.pdf

GeoTiff. (2005) GeoTIFF FAQ Version 2.3. Retrieved April 14, 2010 from http://www.remotesensing.org/geotiff/faq.html

Google. (2010a). "Google Earth: Earth Gallery." Retrieved 13 May 2010 from Google official site. http://www.google.com/gadgets/directory?synd=earth&cat=featured&preview=on

Google. (2010b). "New in Earth 5." Retrieved 17 May 2010 from Google Earth official site. http://earth.google.com/tour.html#v=1

Hittner, Brian. (2003). "Rendering large-scale terrain models and positioning objects in relation to 3D terrain." Master's thesis, Naval Postgraduate School.

ImageMagick. (n,d.). "Introduction to ImageMagick" Retrieved 14 April 2010 from http://www.imagemagick.org/script/index.php

JPEG 2000. (n.d.). "What is JPEG 2000?" Retrieved 12 April 2010 from JPEG 2000 Official Site. http://www.jpeg.org/jpeg2000

Leaver, Greg. (1993). "VRML terrain modeling for the Monterey Bay National Marine Sanctuary (MBNMS)." Master's thesis, Naval Postgraduate School.

NASA. (n.d.) "History of Blue Marble." Retrieved 29 June 2010 from NASA official site. http://earthobservatory.nasa.gov/Features/BlueMarble/BlueMarble_history.php

NASA. (2010). World Wind Wiki. Retrieved 17 May 2010 from Nasa World Wind Central site. http://worldwindcentral.com/wiki/World_Wind.

National Geospatial-Intelligence Agency. (2008). "Controlled image base." Retrieved 19 May 2010, from NGA official site. https://www1.nga.mil/ProductsServices/TopographicalTerrestrial/ControlledImageBase/Pages/default.aspx

National Geospatial-Intelligence Agency. (n.d.). "DNC Background," Retrieved 26 April 2010. from the National Geospatial-Intelligence official site. http://www.nga.mil/NGAPortal/DNC.portal?_nfpb=true&_pageLabel=dnc_portal_page_63.

Naval Postgraduate School. (2010). "Cluster & Resources." Retrieved 7 April 2010 from NPS official site http://www.nps.edu/Technology/HPC/ClusterResources.html

Norbraten, T. (2009). “ImageConverter” from Project Planet Earth source code source http://planet-earth.svn.sourceforge.net/viewvc/planet-earth/RezDir/rez/src/imageutils/ImageConverter.java

OGC. (2010). “KML.” from KML Standards Web site. http://www.opengeospatial.org/standards/kml/

SAVAGE. (2008). “Basic, Geospatial: Hello Earth.” Retrieved 13 May 2010 from X3D Basic Examples site. http://www.web3d.org/x3d/content/examples/Basic/GeoSpatial/_pages/page11.html.

SAVAGE. (2010). “Savage MMOG Hudson build server.” Retrieved 2 August 2010. http://mmog.ern.nps.edu/hudson/

Stockli, R., Vermote, E., Saleous, N., Simmon, R. & Herring, D. (2005). “The Blue Marble Next Generation - A true color earthdataset including seasonal dynamics from MODIS.” Retrieved 25 April 2010 from NASA official site. http://earthobservatory.nasa.gov/Features/BlueMarble/bmng.pdf

Thorne, C. (2007a). “Origin-centric techniques for optimising scalability and the fidelity of motion, interaction and rendering,” Thesis, University of Western Australia, School of Computer Science and Software Engineering.

Thorne, C. (2007b). “Rez,” Retrieved 26 April 2010 from Rez official site. http://www.rez3d.com

Web3D.   (n.d.a) “About the Web3D Consortium.” Retrieved 23 June 2010 from Web3d official site.   http://www.web3d.org/about

Web3D.   (n.d.b) “CAD Working Group.” Retrieved 28 June 2010 from Web3d official site.   http://web3d.org/x3d/workgroups/cad

Web3D.   (2008). “GeoSpatial Component Specification,” Retrieved 5 April 2010 from X3D Specification ISO/IEC 19775-1:2008. http://www.web3d.org/x3d/specifications/ISO-IEC-19775-1.2-X3D-AbstractSpecification/Part01/components/geodata.html

Web3D. (2010). “X3D and HTML5.” Retrieved 28 June 2010 from Web3d official site. http://www.web3d.org/x3d/wiki/index.php/X3D_and_HTML5

Web3D (2002). “GeoVRML.org.” Retrieved 28 June 2010 from http://www.ai.sri.com/geovrml

X3D-Earth. (2007). “X3D-Earth Globe Under Development.” Retrieved 2 July 2010 from X3D-Earth Official Web site.   http://x3d-earth.nps.edu

Yoo, B., & Brutzman, D. (2009). “X3D-Earth Terrain-Tile Production Chain for Georeferenced Simulation,” *Proceedings of the 14th international conference on 3D Web technology*, Los Angeles, CA: ACM.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Don Brutzman
   Naval Postgraduate School
   Monterey, California

4. Don McGregor
   Naval Postgraduate School
   Monterey, California

5. Terry Norbraten
   Naval Postgraduate School
   Monterey, California

6. Aaron Marks
   Dynamis, Inc.

7. Johannes Behr
   Web3D
   Fraunhofer, Germany

8. Laura Moore
   NGA
   Bethesda, Maryland

9. Rex Melton
   Yumatech
   Seattle, Washington

10. Byounghyun Yoo
    MIT, Singapore
    Singapore

11. Mike McCann
    MBARI
    Moss Landing, California

12. David Colleen
Planet 9 Studios
Palo Alto, California

13. Jeff Haferman
Naval Postgraduate School
Monterey, California

14. Eric Adint
Naval Postgraduate School
Monterey, California

15. Jeff Haferman
Naval Postgraduate School
Monterey, California

16. Alan Hudson
Yumatech
Seattle, Washington

17. Christine Haska
Naval Postgraduate School
Monterey, California

18. Todd Hughes
DARPA
Arlington, Virginia

19. Joe LoPiccolo
Naval Postgraduate School
Monterey, California

20. John Moore
Navy Modeling Simulation Office
Washington, D.C.

21. Mark Pullen
George Mason University
Washington, D.C.

22. Peter Schickel
Bit Management
Berg, Germany

23. J. A. Stewart
Communication Research Center Canada
Ontario, Ontario, Canada

24. Alex Viana
Naval Facilities Engineering
Washington, D.C.

25. Audrey Marks
USDA
Washington, D.C.